Department of Computer Science

Virtual Machine Management for Efficient Cloud Data Centers with Applications to Big Data Analytics

Nguyen Trung Hieu

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 140/2016

Virtual Machine Management for Efficient Cloud Data Centers with Applications to Big Data Analytics

Nguyen Trung Hieu

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on 31 August 2016 at 12 noon.

Aalto University
School of Science
Department of Computer Science
Distributed Systems, Mobile Computing and Security

Supervising professor
Assistant Professor Mario Di Francesco, Aalto University, Finland

Thesis advisor
Assistant Professor Mario Di Francesco, Aalto University, Finland

Preliminary examiners
Associate Professor Adlen Ksentini, University of Rennes 1, France
Associate Professor Dijiang Huang, Arizona State University, USA

Opponent
Assistant Professor Hong-Linh Truong, TU Wien, Austria

Aalto University publication series
DOCTORAL DISSERTATIONS 140/2016

ISBN 978-952-60-6913-5 (printed)
ISBN 978-952-60-6912-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-6912-8

Unigrafia Oy
Helsinki 2016

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Nguyen Trung Hieu

Name of the doctoral dissertation
Virtual Machine Management for Efficient Cloud Data Centers with Applications to Big Data Analytics

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 140/2016

Field of research Computer Science and Engineering

Manuscript submitted 19 January 2016 **Date of the defence** 31 August 2016

Permission to publish granted (date) 15 June 2016 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Infrastructure-as-a-Service (IaaS) cloud data centers offer computing resources in the form of virtual machine (VM) instances as a service over the Internet. This allows cloud users to lease and manage computing resources based on the pay-as-you-go model. In such a scenario, the cloud users run their applications on the most appropriate VM instances and pay for the actual resources that are used. To support the growing service demands of end users, cloud providers are now building an increasing number of large-scale IaaS cloud data centers, consisting of many thousands of heterogeneous servers. The ever increasing heterogeneity of both servers and VMs requires efficient management to balance the load in the data centers and, more importantly, to reduce the energy consumption due to underutilized physical servers. To achieve these goals, the key aspect is to eliminate inefficiencies while using computing resources. This dissertation investigates the VM management problem for efficient IaaS cloud data centers. In particular, it considers VM placement and VM consolidation to achieve effective load balancing and energy efficiency in cloud infrastructures. VM placement allows cloud providers to allocate a set of requested or migrating VMs onto physical servers with the goal to balance the load or minimize the number of active servers. While addressing the VM placement problem is important, VM consolidation is even more important to enable continuous reorganization of already-placed VMs on the least number of servers. It helps create idle servers during periods of low resource utilization by taking advantage of live VM migration provided by virtualization technologies. Energy consumption is then reduced by dynamically switching idle servers into a power saving state. As VM migrations and server switches consume additional energy, the frequency of VM migrations and server switches needs to be limited as well. This dissertation concludes with a sample application of distributed computing to big data analytics.

Keywords Virtual Machine (VM) consolidation, VM placement, VM migration, Multiple resource prediction, Data centers, Cloud computing, Big data analytics

ISBN (printed) 978-952-60-6913-5 **ISBN (pdf)** 978-952-60-6912-8

ISSN-L 1799-4934 **ISSN (printed)** 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2016

Pages 154 **urn** http://urn.fi/URN:ISBN:978-952-60-6912-8

Preface

This work has been carried out between September 2012 and August 2016 with the Distributed Systems, Mobile Computing and Security group (formerly Data Communication Software) at the Department of Computer Science, Aalto University School of Science, Finland. This doctoral dissertation would not have been completed without the support and guidance of a number of people and organizations during my studies toward a doctoral degree.

First of all, I would like to address special thanks to my former supervisor, Professor Antti Ylä-Jääski, who has given me the opportunity to undertake a PhD and financially supported the first part of my doctoral studies in the Department of Computer Science at Aalto University. I am grateful to Professor Mario Di Francesco, who was my instructor from the beginning of my doctoral studies and was appointed as my supervisor in September 2013. His expertise, professional conduct, and devotion to research allowed me to complete my doctoral degree. I would also like to thank Professor Sangtae Ha for hosting my research visit to the University of Colorado at Boulder, USA, between November 2015 and March 2016.

I would like to thank other professors in the Department of Computer Science, Aalto University, for their profound knowledge, fascinating courses, and fruitful discussion during my doctoral studies.

I would like to sincerely thank Professor Adlen Ksentini and Professor Dijiang Huang, who served as the official preliminary examiners of this dissertation. I would also like to extend my gratitude to the dissertation opponent, Professor Hong–Linh Truong from TU Wien. Their valuable comments helped me improve the quality of this dissertation.

I also would like to extend my gratitude and thanks to the department secretaries and laboratory managers for supporting and creating

an excellent working environment. I would like to thank Laura Kuusisto-Noponen, Maarit Vuorio, Kristiina Hallaselkä, Emma Holmlund, Katri Seitsonen and Jaakko Kotimäki for their support during my doctoral studies. It was because of their sincere help that I was able to concentrate on research.

Many thanks to all the past and current members of the Distributed Systems, Mobile Computing and Security group at the Aalto University School of Science. In particular, I thank Professor Tuomas Aura, Sanja Scepanovic, Pranvera Kortoçi, Vu Ba Tien Dung, Vu Hoang Nam and Ming Li for their friendship and help during my doctoral studies. I also thank my Vietnamese friends for sharing not only happiness but also difficulty in my life over several years abroad.

I appreciate financial support from the Academy of Finland, the Helsinki Doctoral Education Network in Information and Communications Technology (HICT), the Flexible Spaces Services activity of the EIT ICT labs, the Ulla Tuominen Foundation, and the Google Inc.

I am always thankful to my parents as well as my younger sister and brother, for their endless love, unconditional support, and encouragement during my studies. I thank my wife Cao Hoang Thanh Nha for her love, inspiration, patience, and for making my life filled with happiness. They always wanted me to achieve this goal and supported me in every possible way. I am dedicating this dissertation to them.

Espoo, July 6, 2016,

Nguyen Trung Hieu

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Nguyen Trung Hieu, Mario Di Francesco and Antti Ylä-Jääski. A Virtual Machine Placement Algorithm for Balanced Resource Utilization in Cloud Data Centers. In *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD)*, Anchorage, Alaska, USA, pages 474-481. DOI: 10.1109/CLOUD.2014.70, 27 June - 2 July 2014.

II Nguyen Trung Hieu, Mario Di Francesco and Antti Ylä-Jääski. A Multi–Resource Selection Scheme for Virtual Machine Consolidation in Cloud Data Centers. In *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, pages 234-239. DOI:10.1109/CloudCom.2014.130, September 15-18 2014.

III Nguyen Trung Hieu, Mario Di Francesco and Antti Ylä-Jääski. Virtual Machine Consolidation with Usage Prediction for Energy–Efficient Cloud Data Centers. In *Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD)*, New York, USA, pages 750-757. DOI:10.1109/Cloud.2015.104, 27 June - 2 July 2015.

IV Nguyen Trung Hieu, Mario Di Francesco and Antti Ylä-Jääski. Virtual Machine Consolidation with Multiple Usage Prediction for Energy–Efficient Cloud Data Centers. *IEEE Transactions on Services Computing*, Under review, 14 pages, March 2016.

V Nguyen Trung Hieu, Mario Di Francesco and Antti Ylä-Jääski. Extracting Knowledge from Wikipedia Articles through Distributed Semantic Analysis. In *Proceedings of the 13th ACM International Conference on Knowledge Management and Knowledge Technologies (i-KNOW)*, Graz, Austria, pages 188-195. DOI:10.1145/2494188.2494195, September 04-06 2013.

Author's Contribution

Publication I: "A Virtual Machine Placement Algorithm for Balanced Resource Utilization in Cloud Data Centers"

The author of this dissertation is the primary contributor of the publication. He proposed the original idea of balancing resource utilization, designed the virtual machine placement algorithm and performed the evaluation by simulation.

Publication II: "A Multi–Resource Selection Scheme for Virtual Machine Consolidation in Cloud Data Centers"

The author of this dissertation is the primary author of this publication. He proposed the multiple resource selection scheme, designed the balanced multiple–resource utilization algorithm and performed the evaluation by simulation.

Publication III: "Virtual Machine Consolidation with Usage Prediction for Energy–Efficient Cloud Data Centers"

The author of this dissertation is the primary author of the publication. He proposed the original idea for virtual machine consolidation with usage prediction. He designed an efficient usage prediction approach, designed the consolidation algorithm and performed the evaluation by simulation.

Publication IV: "Virtual Machine Consolidation with Multiple Usage Prediction for Energy–Efficient Cloud Data Centers"

The author of this dissertation is the primary author of this publication. He proposed the original idea and the corresponding problem formulation, designed the multiple usage prediction as well as the virtual machine consolidation algorithm with prediction. He also performed the evaluation by simulation.

Publication V: "Extracting Knowledge from Wikipedia Articles through Distributed Semantic Analysis"

The author of this dissertation is the primary author of this publication. He proposed the original idea of using distributed computing for fast processing, designed the semantic relatedness metric and performed the experimental evaluation.

List of Abbreviations

IaaS	Infrastructure–as–a–Service
VM(s)	Virtual Machine(s)
VMM	Virtual Machine Manager
CPU	Center Processing Unit
QoS	Quality of Service
SLA	Service Level Agreement
RQ(s)	Research Question(s)
I/O	Input or Output
OS	Operating System
KVM	Kernel–based Virtual Machine
FIFO	First In First Out
GCD	Google Cluster Data
U	Resource Utilization
$\hat{U}$	Average Resource Utilization
B	Resource Balance
$\hat{B}$	Average Resource Balance
RH	Resource Hottest
RT	Resource Temperature
RC	Resource Correlation
Max–BRU	Maximized and Balanced Resource Utilization
MRS	Multiple Resource Selection
BRMU	Balanced Multiple Resource Utilization
UP	Usage Prediction
VMCUP	VM Consolidation with Usage Prediction
MUP	Multiple Usage Prediction
VMCUP–M	VM Consolidation with Multiple Usage Prediction
OHD–MUP	Overloaded Host Detection with Multiple Usage Prediction
UHD–MUP	Underloaded Host Detection with Multiple Usage Prediction

THR	Static Threshold
THR–MUP	Static Threshold with MUP
MAD	Median Absolute Deviation
IQR	Interquartile Range
LR	Local Regression
LR–MUP	Local Regression with MUP
FF	First–Fit
BF	Best–Fit
WF	Worst–Fit
NF	Next–Fit
FFD	First–Fit Decreasing
PABFD	Power–aware Best Fit Decreasing
PABFD–MUP	Power–aware Best Fit Decreasing with MUP
DRR	Dynamic Round Robin
MRT	Minimum Resource Temperature
MMT	Minimum Migration Time
MC	Maximum Correlation
MU	Minimum Utilization
RS	Random Selection
BG	Black–box and Gray–box
BG–MUP	Black–box and Gray–box with MUP
VSR	Volume–to–Size Ratio
RRV	Resource Requirement Vector
TCV	Total Resource Capacity Vector
UCV	Utilized Capacity Vector
BFVD	Best–Fit VectorDot
FFVD	First–Fit VectorDot
WFVD	Worst–Fit VectorDot
RBFVD	RelaxedBest–Fit VectorDot
MM	Market Mechanism
LAJ	Latest Arrival Job
BL	Backfill Lowest
BB	Backfill Balance
CPULoad	CPU Load–aware
mPP	Min Power Parity
mPPH	Min Power Placement with History
pMaP	Balance between Power and Migration Cost
DSA	Distributed Semantic Analysis

WSRel	Word Semantic Relatedness
TF–IDF	Term Frequency – Inverse Document Frequency
ESA	Explicit Semantic Analysis
TSA	Temporal Semantic Analysis
LSA	Latent Semantic Analysis
LSAC	LSA@CU Boulder
M&C	The Miller and Charles' dataset
R&G	The Rubenstein and Goodenough's dataset
WS–353	The Finkelstein et al.'s dataset (WordSimilarity–353)

List of Tables

List of Figures

1. Introduction and Motivation

Infrastructure–as–a–Service (IaaS) cloud data centers — including Amazon EC2 [3], IBM Cloud [61], Google Compute Engine [47], and Rackspace [116] — offer several types of virtual machines (VMs) that differ in their amount of resources based on the pay–as–you–go model [21, 138]. This allows cloud users to run their applications on the most appropriate VM instances and pay for the actual resources that are used [2]. To support the growing service demands of their users, cloud providers have recently begun to deploy an increasing number of large–scale IaaS cloud data centers, thus resulting in a huge energy consumption [91]. As reported in Analytics Press, energy consumption by data centers worldwide increased by about 56% from 2005 to 2010, and in 2010 likely accounted for between 1.1% and 1.5% of the total electricity use [70]. Additionally, the energy–related costs accounted for roughly 42% of the total costs of a data center [54].

IaaS cloud data centers currently consist of many thousands or even millions of heterogeneous servers and each server may host a set of heterogeneous VMs. Accordingly, Rackspace's IaaS has increased the total server count in the third quarter of 2014 to 110,453, up from 107,657 servers at the end of the previous quarter, and the number of servers continuously grows [69, 115]. Amazon EC2 had approximately 40,000 servers and launched 80,000 VMs daily in 2011 [33] and it has been estimated that one and half million servers were running millions of VMs in 2014 [136]. Google was estimated to have around 1.8 million servers as of January 2012 and 2.3 million servers by early 2013 [63, 123]. Furthermore, the number of VM requests deployed in a cloud data center each day can be very large; it has been estimated that approximately 360,000 VM requests were deployed within 24 hours in a single data center in 2013 [9]. These numbers may be even larger today as cloud computing

is much more popular than two years ago. Therefore, the ever increasing heterogeneity for both the physical servers and the VMs needs to be managed efficiently in order to achieve the following key goals: maximize resource utilization and reduce the energy costs [24, 58, 134, 150].

To address the problem of high energy use in IaaS cloud data centers, it is necessary to eliminate inefficiencies while using computing resources. This may be achieved by improving resource allocation and management [5, 74, 94, 103, 114]. However, such resources may also vary over time due to dynamic workloads that require resizing, creating, and (or) terminating VMs. Furthermore, computing resources consist of multiple types (or *dimensions*) — including CPU, memory, disk, and network bandwidth — and all need to be considered while designing energy–efficient mechanisms for resource management [149]. As a consequence, if the owners of cloud data centers could not effectively schedule and reallocate heterogeneous VM instances and resource types, some hosts might become overloaded while other hosts might be underutilized. Eventually, such an unbalanced use of hosts could result in unnecessary activation of servers, thus consuming huge amounts of electrical energy and resulting in high operating costs [50, 87]. Moreover, by considering the actual VM resource utilization after VM placement, increasing the workload of some already–placed VMs may cause the corresponding physical servers to be overloaded, possibly affecting the quality of service (QoS) experienced by the hosted applications. In fact, the QoS level offered to cloud users needs to fulfill the service level agreement (SLA) of the cloud provider [11, 17]. On the other hand, physical servers may become underloaded due to a decrease in the VM workloads, but would still contribute to significant amounts of power consumption in data centers. In such a scenario, it is beneficial to move all VMs to other servers and switch the underloaded machine into a power–saving state (e.g., suspend) to save energy [16, 90].

One method to improve resource utilization and reduce the energy consumption is VM placement [59, 78, 79, 83]. In most scenarios, when cloud users submit their VM requests, some physical server(s) in the IaaS cloud data center will be selected to deploy the required VMs [82, 102]. Particularly, VM placement allows cloud providers to allocate a set of VMs to physical servers with the goal to minimize the number of active machines to accommodate the VMs [52, 60, 80, 97]. To this regard, choosing the most appropriate target machine in a large pool of physical servers to create the requested VMs provides a strong motivation for cloud providers to

maximize their operational efficiency [65].

While addressing the VM placement problem is important to minimize the number of active servers starting from the VM submission, VM consolidation enabled by virtualization technologies [99, 122] is even more important to support continuous consolidation of already–placed VMs on the least number of physical servers [29, 35, 89, 110]. Virtualization allows multiple VMs to be placed into the same physical server and each VM may run multiple application tasks, thus ensuring that the server is optimally utilized while reducing the energy consumption [76, 101, 126]. Multiple resource types — i.e., CPU, memory, storage, and network bandwidth — may be dynamically provisioned for a VM according to the current resource requirements [1, 102]. This enables the consolidation of VMs in the minimum number of servers to switch off unused machines, with the goal to reduce the total power consumption [12, 104, 145]. By taking advantage of virtualization technologies, cloud providers are allowed to increase the energy efficiency of the cloud data centers and scale the costs of the offered virtualized resources.

Another capability provided by virtualization is live migration, which is the ability to transfer a VM between physical servers with little or no migration downtime during the process [27, 56, 142]. By using live migration, VMs may be dynamically consolidated into a few physical servers, then unused machines (i.e., those that do not host any VMs) may be switched off [12, 36, 80, 144, 145]. This approach helps improve the resource utilization and allows energy savings in compliance with the SLA [49]. VM consolidation with live migration is closely related to the problem of: (1) determining when a server is overloaded (i.e., a *hot spot*), then migrating the potential VMs from such a server to maintain a certain QoS; and (2) determining when a server is underloaded (i.e., a *cold spot*), then migrating all VMs from such a server to minimize energy consumption. Idle hosts are automatically switched to a low–power mode to reduce the energy consumption. When required, the low–power hosts are reactivated to accommodate newly–created VMs or VMs being migrated.

However, it is challenging to decide whether a host is overloaded or underutilized due to the diverse set of user applications and the variability of the VM workloads with time, especially in a cloud data center with millions of machines. Purely based on the last observed utilization for decision making, existing solutions may cause unnecessary migrations, thus increasing the overhead: the energy for VM migration, the performance

degradation of the hosted applications, and extra network communications [37, 88, 125, 147, 148]. For those reasons, even though live migration is a suitable solution for managing VM populations, it is important to avoid unnecessary VM migrations. For instance, commercial IaaS platforms such as Amazon EC2 and Microsoft Azure do not use VM migration at all. In any case, hot and cold spots should be carefully determined, in order to limit the frequency of VM migrations. During migration of VMs, if there is no active physical server with sufficient resources available, an inactive server is automatically started and the selected VMs are allocated to such a machine. In addition, when a host is underutilized, all VMs from such a host are selected for migration if they can be consolidated into other hosts without causing overutilization. Idle servers are then switched to a low–power state to save energy. However, switching the power state of a host from idle to a low–power state and vice versa consumes additional energy [50, 51, 71, 85, 133]. Therefore, as VM migrations and server switches are essential for power reduction, it is even more important to avoid massive migrations and limit power state switches.

This doctoral dissertation is motivated by the limitations of the current VM management algorithms implemented in the OpenStack, OpenNebula, and Eucalyptus cloud management middlewares [57, 107, 113]. Such IaaS platforms use very simple VM management algorithms, i.e., round robin, greedy First–Fit, load–aware, and do not enable energy–efficient cloud infrastructures. Therefore, VM management solutions for large–scale data centers must be designed to effectively take power consumption into account while guaranteeing the application QoS level. Accordingly, the scientific contribution of this dissertation is VM management for energy–efficient IaaS cloud data centers. Particularly, to achieve energy efficiency in cloud infrastructures, two key VM management algorithms — specifically, for VM placement and VM consolidation — are proposed. They are capable of: (1) limiting the number of active physical servers; (2) creating idle servers, then transitioning such idle servers in a power–saving state and reactivating them once required; and (3) minimizing the number of VM migrations and server switches along with the number of activated servers. Besides, big data applications run on large clusters within data centers, and the related energy costs make their processing time an extremely critical component. MapReduce [30] and its open–source implementation Hadoop [39] have emerged as the leading computing platforms for big data analytics. Reducing the processing time

of big data applications based on Hadoop MapReduce leads to a significant decrease in the overall operating cost of a data center. This dissertation further addresses data–intensive applications by designing a distributed computing system for big data analytics, specifically, for semantic analysis. The proposed system helps reduce the execution time for running data–intensive processes, thus improving the system performance and the energy consumption as a side effect.

1.1 Research Problems and Questions

This dissertation tackles the above–mentioned research challenges by proposing algorithms for managing VMs with the goal to optimize the performance of an IaaS cloud data center in terms of balanced use of resources and reduced energy consumption. In particular, the following research problems are considered:

- **Energy Efficiency**. Energy efficiency is a major concern for IaaS cloud data center providers. By taking advantage of virtualization technologies, cloud providers are allowed to increase the energy efficiency of their infrastructures by minimizing the number of active servers along with the number of VM migrations. Furthermore, energy saving may be achieved by switching idle servers to a power–saving state while minimizing the number of power state changes (i.e., between on and off). In large–scale cloud data centers, improving the energy efficiency only by a few percent may save millions of dollars in electricity costs [67].

- **Scalability**. IaaS cloud data centers to date consist of millions of heterogeneous physical servers and VMs. To improve the utilization of physical resources, the management of IaaS clouds would ideally employ optimal VM placement and VM consolidation, which are known to be NP hard problems [53, 129]. Thus, a practical management of such large–scale data centers requires highly scalable VM management algorithms. Consequently, VM placement and consolidation algorithms dealing with an ever growing number of heterogeneous servers and VMs are a challenge with a high level of complexity.

- **Performance**. In IaaS clouds, heterogeneous resources such as computation and storage are provisioned on–demand by cloud providers in the

form of VMs. This allows cloud users to run their applications on the most appropriate VMs and pay for the actual resources that are used. Specifically, IaaS clouds are suitable for deploying high performance computing [32, 100], scientific workflows [121], and social network applications [25]. Furthermore, VM management algorithms should guarantee adequate performance by avoiding overloads of any resources under VM placement and VM consolidation.

In more detail, the following research questions (RQs) related to the previous research problems are investigated and answered in this dissertation:

- **RQ1: How to consider multiple types of resources and balance the load among them?** VM management for energy–efficient IaaS cloud data centers should take into account multiple resource types simultaneously, since CPU is not the only critical resource in cloud data centers. In fact, also memory and network bandwidth may become a bottleneck, possibly causing violations in the SLA. Furthermore, multiple applications of various types may have different demands in terms of resources. For instance, a request for computer–intensive applications (e.g., weather forecast and big data analytics) needs more CPU or memory resources and a request for database and memory caching applications (e.g., web hosting and online banking) needs more I/O resources. Therefore, VM management algorithms should consider multiple types of resources and spread the load among them to help resolve one of the most compelling issues in cloud data centers: underutilization of physical servers due to an unbalanced use of resources across multiple dimensions.

- **RQ2: When to migrate virtual machines?** The VM consolidation problem consists of two basic phases: (1) determining when a server is overloaded, then migrating the potential VMs from such a server to maintain a certain QoS; and (2) determining when a server is underloaded, then migrating all VMs from such a server to minimize energy consumption. In a large IaaS cloud data center with millions of physical machines, the variability of already–placed VM workloads with time makes challenging to decide whether a host is overloaded or underutilized. Consequently, more efficient overload and underload management

schemes are needed to correctly make decisions on VM migration. In other words, hot and cold spots should be reliably determined across multiple resources to limit the frequency of VM migrations.

- **RQ3: Which and how many virtual machines should be selected for migration?** When a server is overloaded, it is challenging to determine which and how many VMs should be selected for migration to suitable hosts. As migration is expensive, VM selection plays an important part in limiting the number of VMs migrations. The problem consists of selecting one or more potential VMs for migration to reduce the resource load of the considered servers.

- **RQ4: Where to place a virtual machine just created or under migration?** The target physical server should be correctly selected in large cloud data centers to allocate newly–created VMs or those under migration. As the result of VM allocation requests or migrations, the target servers may become overloaded during periods of high resource utilization. Consequently, an overloaded host management scheme is required to detect overload situations. Such a scheme also supports VM placement algorithms in deciding which physical machines are the most suitable to accommodate VMs being submitted or migrated. The problem here consists of selecting a target server not only based on the least increased power consumption but also based on its utilization stability. In other words, a selected server should not be overloaded in the future period of time after placing VMs.

- **RQ5: When and how many servers should be switched to a low–power state and vice versa?** Switching the power state of a host from idle to off or from inactive to on consumes additional energy. In a dynamic environment such as cloud data centers where the resource needs of VMs vary over time, power state switches are essential for reducing energy consumption. However, it is even more important to limit the switching frequency.

- **RQ6: How to design a distributed–based approach for big data applications?** To provide a scalable and efficient approach to processing large amounts of data, it is necessary to study the performance and the energy efficiency of cloud computing jobs in terms of computation

time.

To answer the research questions mentioned above, this dissertation develops a set of algorithms for VM placement (Publication I) and VM consolidation (Publication II, Publication III, and Publication IV). In addition, this dissertation implements a scientific application to perform computationally–heavy tasks (Publication V). The research questions are answered in these publications as shown in Table 1.1.

	Publications	**I**	**II**	**III**	**IV**	**V**
		Energy Efficiency and Scalability				Performance
RQ1:	How to consider multiple types of resources and balance the load among them?	√	√	–	√	–
RQ2:	When to migrate virtual machines?	–	√	√	√	–
RQ3:	Which and how many virtual machines should be selected for migration?	–	√	√	√	–
RQ4:	Where to place a virtual machine just created or under migration?	√	√	√	√	–
RQ5:	When and how many servers should be switched to a low–power state and vice versa?	–	–	√	√	–
RQ6:	How to design a distributed–based approach for big data applications?	–	–	–	–	√

Table 1.1. How the publications address the research questions.

1.2 Methodology

This doctoral dissertation tackles the previously–described research questions by employing the following methods and tools.

- **Infrastructure–as–a–Service (IaaS)** allows customers (e.g., cloud users) to lease and manage virtual resources (e.g., CPU, memory, storage, and network bandwidth) over the Internet in the form of VM instances [13, 151]. Some well–known public IaaS clouds include Amazon EC2 [3], Google Compute Engine [47], and Rackspace [116]. Moreover, a number of open–source IaaS cloud management systems have been developed including CloudStack [40], Eucalyptus [57, 105], Nimbus [106], OpenNebula [107], and OpenStack [113]. This doctoral dissertation mainly focuses on the IaaS model.

- **Virtualization** is widely deployed in large–scale data centers and becomes the foundation of cloud computing due to its ability to isolate

co–located application workloads and its efficiency for resource multiplexing [146]. This technology allows IaaS infrastructures to create several VMs on a physical server; therefore, it reduces the amount of hardware in use and improves the utilization of resources. The virtualization layer for hypervisor–based virtualization is placed between the hardware and the operating system (OS). It is implemented by a Virtual Machine Monitor (VMM) [122], which controls resource multiplexing and manages the allocation of physical resources (e.g., CPU, memory, storage, and I/O devices) to the VMs. There are two major types of implementation of hypervisor–based virtualization [140]: full virtualization (e.g., VMware Workstation [141], VirtualBox [108], Kernel–based Virtual Machine (KVM) [66], and Microsoft Hyper–V [93]) and paravirtualization (e.g., Xen Paravirtualization [6]). This doctoral dissertation focuses on hypervisor–based virtualization; therefore, the term virtualization throughout the dissertation refers to this category.

- **Virtual machine placement** is the process of selecting the appropriate physical server in large cloud data centers to accommodate newly–created or migrated VMs. The goal of VM placement is to assign the VMs to servers in such a way that the number of used servers is minimized.

- **Virtual machine consolidation** is enabled by virtualization technologies [55, 71]. This allows cloud providers to create multiple VM instances on a single physical server, thus improving resource utilization and creating idle servers. The reduction in energy consumption is achieved by switching such idle hosts to low–power states (e.g., suspend, sleep, hibernation, or shutdown) during periods of low utilization, thus eliminating idle power consumption.

- **Live virtual machine migration** is a method to transfer VMs between physical hosts over a local or wide area network without shutting these VMs down [120, 31, 148]. VM migration may be performed automatically by a cloud management system. For instance, multiple VMs can be consolidated on a fewer number of servers for energy saving purposes. However, this process may impact on the performance of applications running on a VM, the source and destination hosts, other VMs, and the network during a migration [130]. VM live migration can be categorized

into three approaches: pre–copy, post–copy, and a combination of both. This doctoral dissertation primarily focuses on the pre–copy approach to live VM migration, which is the chosen method for performing live migration in the Xen hypervisor [27].

- **Multiple usage prediction** utilizes machine learning techniques (e.g., neural networks and linear regression) to predict future resource utilization in the cloud with respect to time [62]. However, training neural network models takes significant time, which depends on the size of the input as well as the frequency of predictions. Therefore, it is important to determine effective learning algorithms for consolidating VMs in dynamic environments, such as cloud data centers with millions of heterogeneous machines and heterogeneous resource types. This dissertation utilizes the Multiple Linear Regression technique [143] to forecast the future resource utilization in terms of multiple resource types — i.e., CPU, memory, storage, and network bandwidth — based on historical data.

- **CloudSim** is the simulation tool used to evaluate the effectiveness of the proposed schemes in a practical cloud scenario [22]. This dissertation extends CloudSim to handle both multi–resource types and energy–awareness.

- **Virtual machine utilization** follows the workload traces from both real–world public workloads as well as synthetic workloads with different availability of resource types, including PlanetLab VMs (CPU and memory) [111], Google Cluster Data (CPU and memory) [137], the Real Parallel Workloads from Production Systems (CPU and memory) [119], Amazon EC2 (CPU, memory, and storage), and the uniform and normal distribution (CPU, memory, storage, and network bandwidth) [12, 82]. Specifically, the usage of each type of resources in the real–world public dataset is collected every five minutes based on an empirical evaluation of the considered workloads (additional considerations on this aspect are provided in Chapter 5). According to the available resource types of the considered workloads, this doctoral dissertation sets the resource dimension to $D = 2$, $D = 3$, and $D = 4$, then adopts $|D|$–dimensional VM placement and VM consolidation in the simulations.

- **Hadoop MapReduce** is a software framework which allows cloud users to solve computationally–intensive problems. In particular, MapReduce is a simple programming model to run distributed computation on very large data sets using large clusters of commodity machines [30]. Its open–source implementation Hadoop [39] includes two main components: the Hadoop Distributed File System (HDFS) and the MapReduce distributed computing paradigm.

1.3 Contributions

This dissertation summarizes five peer reviewed publications whose contribution is briefly described below.

Publication I proposes a multi–resource VM placement algorithm for maximizing the utilization and balancing the load across different types of resources in cloud data centers. The research goal is to allocate a set of VMs to physical machines such that the number of servers required to accommodate the VMs is minimized. This is achieved by implementing a multi–resource VM placement algorithm. However, most of the existing solutions only consider a limited number of resource types (in many cases only the CPU), thus resulting in an unbalanced load or in the unnecessary activation of physical servers. To solve this limitation, Publication I investigates the use of multiple resource–constraint metrics that help find the most suitable server for deploying VMs in large cloud data centers. Accordingly, a multi–resource VM placement algorithm called Max–BRU is proposed for spreading the load across multiple dimensions. Max–BRU is especially attractive for the VM placement problem due to its polynomial time worst–case complexity on the number of the VM deployment requests. The Max–BRU algorithm is evaluated through simulation experiments and is compared with the eight other state of the art approaches: Greedy First–Fit [57, 65, 68, 82, 84], Load–aware [28, 107], VectorDot [125], Market Mechanism [147], the Min–Min and Max–Min heuristics [43, 53, 127], the algorithm proposed in [26], and an extension of the volume metric introduced in [144]. Simulation results demonstrate that Max–BRU obtains a more balanced use of resources than the state of the art. As a consequence, it makes a more efficient use of multiple resources, thus reducing the number of required active physical servers in cloud data centers.

Publication II proposes a multi–resource selection (MRS) scheme for consolidating VMs in cloud data centers. Additionally, an efficient VM consolidation algorithm called BMRU is proposed for balancing the usage of resources across multiple dimensions. The VM consolidation algorithm is important to enable continuous consolidation of already–placed VMs on the least number of physical servers. Therefore, the proposed BMRU algorithm integrates with MRS and uses the current utilization of multiple types of resources to characterize and classify a physical server. This is particularly important to avoid an unbalanced load over multiple types of resources and further increase the resource utilization of a data center. To save energy, the BMRU algorithm detects idle physical servers, transitions them into a power–saving state (e.g., suspend), and reactivates them once required. Before this can be achieved, underloaded host detection and VM migration are performed along with VM consolidation. Both mechanisms aim at placing VMs on the least number of servers and release lightly–utilized machines. To evaluate BMRU, a custom simulator has been written in Java; a 4–dimensional VM consolidation scheme has then been evaluated under synthetic workloads (considered CPU, memory, storage, and network bandwidth) and a 2–dimensional VM consolidation scheme under the real–world Google Cluster Data [137] and RIKEN Integrated Cluster of Clusters [119] workloads (considering CPU and memory). The experimental results have proven that the proposed approach obtains a more balanced use of multiple resources, thus increasing the energy efficiency of a data center by minimizing the number of active physical servers.

Publication III proposes a VM consolidation with usage prediction (VMCUP) algorithm for a more efficient detection of overloaded and underloaded servers to avoid unnecessary VM migrations and server switches. The key idea of the proposed approach is to predict the short–term usage of a single computing resource (i.e., the CPU utilization in a five–minute horizon) and use the current and predicted usage metrics for a reliable characterization of overloaded and underloaded servers. In addition, the proposed solution is aligned with green computing strategies, in which physical machines may be powered off to save energy. The proposed algorithm is evaluated in a practical cloud scenario by extending the CloudSim simulation toolkit [22]. VMCUP is compared to four well–known overutilized host detection approaches in [12]: static threshold (THR), the median absolute deviation (MAD), the interquartile range

(IQR), and a dynamic threshold based on local regression (LR). Extensive experiments performed on the Google Cluster Data [137] and the PlanetLab VMs [111] workload traces show that VM consolidation embedding usage prediction reduces the energy consumption while limiting both the number of migrations and the power state changes. As a consequence, it improves the performance of a data center with a better compliance with the service level agreement (SLA).

Publication IV extends the VMCUP approach presented in Publication III to predict the long–term utilization over multiple types of resources (i.e., CPU, memory, disk, and network bandwidth). This publication makes the following contributions: (1) it adapts the previously proposed usage prediction (UP) in Publication III to enable multiple usage prediction (MUP), in terms of both resource types and the horizon employed to predict future utilization; (2) it introduces a VM selection policy — namely, the minimum resource temperature (MRT) — that migrates a VM based on the joint impact of multiple resources; (3) it embeds the MUP scheme into the power–aware best fit decreasing (PABFD) solution through a new VM placement algorithm called PABFD–MUP so as to select a target physical server not only based on the least increased power consumption but also on its utilization stability, predicted by using the MUP scheme; and (4) it proposes an algorithm for VM consolidation with multiple usage prediction (VMCUP–M) for energy–efficient IaaS cloud data centers. The key idea of the proposed algorithms to achieve both multiple–resource and multiple–step prediction is to apply underloaded and overloaded host management, VM selection, and VM placement under migration. Specifically, the joint use of current and predicted resource utilization over multiple dimensions allows for a reliable characterization of overloaded and underloaded servers, thereby reducing both the load and the power consumption after consolidation. In addition to that, MUP helps limit the number of active servers, the number of VM migrations and power state changes, thus achieving a better compliance with the SLA. VMCUP–M has a polynomial time complexity on the number of the VMs to be allocated in the data center. The CloudSim simulation toolkit [22] is extended to handle multiple types of resources and energy–aware simulations. VMCUP–M is then implemented on top of such an extended version of CloudSim and is evaluated with the following existing approaches in [12]: (1) overutilized host detection: static and dynamic hot thresholds; (2) VM selection: minimum migration time (MMT), maximum correlation

(MC), minimum utilization (MU), and random selection (RS); and (3) VM placement: power–aware best fit decreasing. Furthermore, VMCUP–M is compared to the multiple–resource black–box and gray–box (BG) scheme introduced in [144] through our own implementation based on extending the volume metric across two resources, i.e., CPU and memory. Extensive experiments performed on the 1,600 VMs from Google Cluster Data [137] and the 1,473 VMs from PlanetLab [111] workload traces show that VM consolidation embedding multiple usage prediction reduces the energy consumption while limiting the number of active physical servers, the number of migrations and power state changes, thus increasing the performance of a cloud data center with a better compliance with the SLA.

Publication V proposes Distributed Semantic Analysis (DSA), a big data system that integrates a distributed computing with semantic analysis, thereby allowing cloud users to efficiently process large amounts of data (e.g., huge amounts of Wikipedia articles). This publication is targeted to application–aware approaches that consider high–level application requirements in terms of QoS (e.g., response time) while performing computationally–heavy tasks. In particular, Hadoop MapReduce is used to build an environment that can easily be scaled through virtualization to split the source data and process them in parallel [30, 39]. Extensive experiments are done on a Hadoop MapReduce cluster to determine the execution time for analyzing Wikipedia data and then to evaluate the performance of the proposed DSA system. Accordingly, DSA can significantly improve the energy efficiency of a real IaaS cloud by reducing the computation time to analyze big data.

1.4 Thesis Organization

The rest of this dissertation is organized as follows. Chapter 2 discusses efficient VM placement, while Chapter 3 focuses on VM consolidation for saving energy in cloud data centers. Chapter 4 presents an application of distributed computing to big data analytics. Chapter 5 concludes the dissertation with a summary of the main contributions and a discussion of future research directions. Finally, the original papers are provided at the end of the dissertation.

2. Efficient Virtual Machine Placement

This doctoral dissertation is about efficient Infrastructure–as–a–Service (IaaS) cloud data centers, with focus on energy efficiency. To provide a context for the reader concerning the motivations of the dissertation, this chapter first outlines the key components for energy–efficient cloud data centers. The state of the art on the algorithms for energy–efficient virtual machine (VM) placement is then reviewed. Such a review shows that existing VM placement algorithms mostly consider a limited number of resource dimensions (e.g., a single CPU resource) and have several limitations. They include neglecting multiple types of resources as well as the heterogeneity of physical servers and VMs. To address these limitations, the main contributions in this chapter are to implement and evaluate a VM placement solution which: (1) considers multiple types of resources such as CPU, memory, storage, and network bandwidth; and (2) balances the load across different types of resources while placing VMs.

This chapter presents research done in Publication I. In particular, the Max–BRU VM placement algorithm for maximized balanced resource utilization over multiple dimensions is summarized. The Max–BRU algorithm is evaluated through a custom simulator written in Java. According to the experimental results from both synthetic and real–world workloads, the proposed algorithm significantly reduces the number of active servers, thus minimizing the power consumption of IaaS cloud data centers while, at the same time, balancing the usage of multiple types of resources.

2.1 Case Study: Energy Efficiency of Data Centers

The largest energy consumption within a typical data center is represented by the servers, which account for about 60% of the data center's

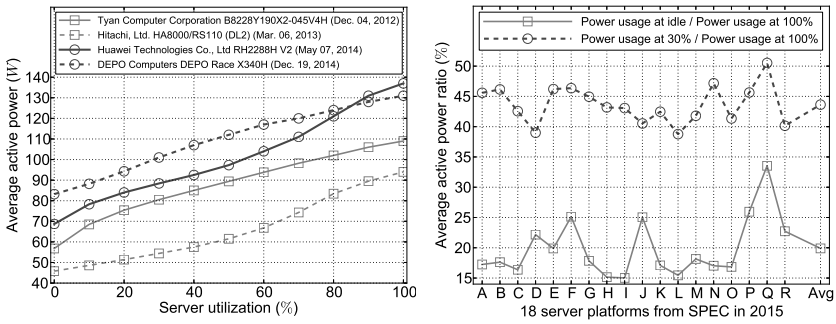
Architecture	Number of servers	CPU	Memory	Server population (%)
1	**6728**	**0.50**	**0.50**	**53.469**
2	**3864**	**0.50**	**0.25**	**30.708**
3	**1003**	**0.50**	**0.75**	**7.971**
4	**795**	**1.00**	**1.00**	**6.318**
5	126	0.25	0.25	1.001
6	54	0.50	0.13	0.429
7	5	0.50	0.03	0.040
8	4	0.50	0.97	0.032
9	3	1.00	0.50	0.024
10	1	0.50	0.06	0.008

Table 2.1. Statistics of machines in the Google Cluster Data.

overall consumption [54]. This means that the energy efficiency of the servers is the key component for an energy–efficient data center. Unfortunately, such servers are heterogeneous and mostly act at low resource utilization levels [7, 8, 44]. To show the heterogeneity of the servers, I first analyze the CPU and memory capacities from the Google Cluster Data dataset [137], consisting of 12,583 machines. The statistics of the servers are shown in Table 2.1, which provides the exact resource capacity of the machines across different types of resources, thereby giving an indication of the heterogeneity of servers in a data center. The data show that physical machines have diverse CPU and memory capacities, specifically, 30.708% of the servers belong to architecture 2 and 7.971% of the servers to architecture 3.

Garraghan et al. [44] presented the average CPU and memory utilization for the top four architectures, highlighted in bold (98.466% of the server population) in Table 2.1. The results confirm that the average CPU and memory utilization over all the servers are below 48% and 50%, respectively. A study by IBM researchers in 2012 [15] reported that the utilization of CPU, memory, and disk were 18%, 78%, and 75% (respectively) over several thousands of servers and a two–year period. Such results confirm that, while the CPU is seemingly underutilized, memory and disk may already operate at high resource utilization levels. In this scenario, it is a challenging problem to improve the resource utilization of only the CPU without considering other resource types.

Underutilization of the servers in IaaS cloud data centers is a primary source of inefficiency. Accordingly, the authors in [8] showed that Google servers operate most of the time at between 10% and 50% of their maximum utilization levels. Such server utilization consumes almost half of the energy compared to full utilization. Furthermore, a server still con-



(a) **(b)**

Figure 2.1. Server power usage at varying utilization levels of server platforms from SPEC [118], i.e., (a) from idle to peak performance and (b) active power ratio at idle and at 30% utilization (relative to the 100% utilization).

sumes up to 60% of its peak power when it is completely idle, [7, 34, 77]. Figure 2.1a illustrates this through the power consumption of four different server platforms from the Standard Performance Evaluation Corporation (SPEC) [118] against their utilization levels (0% means the server is completely idle while 100% means the server is fully utilized). The results indeed demonstrate that the considered servers consumed up to 60% of peak power at 0% utilization. Figure 2.1b shows the relative efficiency of the 18 servers (referred from A to R, Avg is average) in the SPEC power benchmark (data from 2015) when running idle and at 30% utilization compared to the values obtained at a 100% utilization level. The results confirm that idle servers consume about 20% of their peak power on the average and that the servers operating at a low utilization (i.e., 30%) consume around 43% of the peak power on the average. Overall, the study showed that the overall energy consumption of a data center is due to the underutilization of servers. Precisely, the energy is wasted due to the low average resource utilization, typically ranging at between 10% and 50%, where servers have their worst energy efficiency.

The key aspect for energy–efficient IaaS cloud data centers is to improve the efficiency of the servers. This is possible by balancing the usage of resources across multiple dimensions, thus helping to increase server utilization. Furthermore, reducing the number of active physical servers significantly decreases the overall energy consumption. This is achieved by optimizing the assignment of VMs to physical machines in IaaS cloud data centers. Energy consumption is further reduced by dynamically switching

servers to a low–power mode during idle times.

Accordingly, this dissertation specifically focuses on managing VMs for energy–efficient IaaS cloud through two main methods: VM placement and VM consolidation (the details of the latter can be found in Chapter 3). VM placement aims at balancing the load across different types of resources, thus limiting the number of active servers, while VM consolidation enables to create idle servers that can be switched into a power–saving state. Both approaches try to improve the server utilization through more energy–efficient operating modes within a cloud data center.

In the following, the first major contribution of the dissertation with focus on VM placement for efficient cloud data centers (Publication I) is discussed after reviewing the related work on VM placement.

2.2 Virtual Machine Placement

Virtual machine placement is the process of selecting the appropriate physical servers in large cloud data centers to accommodate newly–created or migrated VMs. The goal of VM placement is to assign the VMs to physical servers in such a way to allow efficient usage of resources [14, 26, 60, 86, 125]. Finding the optimal allocation of physical servers has long been known to be an NP hard problem [53, 129]. As a consequence, many heuristic algorithms have been proposed to find a feasible solution within a reasonable time [19]. A well–known mapping algorithm is greedy First–Fit (FF), which is used in the Eucalyptus cloud management middleware [57, 68, 84]. The First–Fit algorithm allocates each VM request to the first physical server which satisfies the demands of the VM for all resources, starting from the first server sorted according to a predefined metric (e.g., available resources or power efficiency). An improvement version of the FF algorithm is called First–Fit–Decreasing (FFD), which sorts the VMs in decreasing order according to their resource demands. FFD uses no more than $11/9 \cdot \mathrm{OPT} + 1$ bins, where OPT is the number of bins provided by the optimal solution [96]. Other greedy algorithms have also been devised to solve the problem of VM placement, including Best–Fit (BF), Worst–Fit (WF), and Next–Fit (NF). Besides, OpenNebula [28, 107] uses a load–aware policy that selects first the physical server with the least used CPU for allocating VM requests. Other heuristic–based resource allocation algorithms are Min–Min and Max–Min, which have been largely adopted in the literature [43, 53, 127] to assign compu-

tation resources in such a way to guarantee a target processing time. As a consequence, the available *CPU capacity* becomes the main allocation criterion affecting the task completion time. The major limitation of these algorithms is that they only focus on meeting one objective called utility of resources (greedy and load–aware) and on managing a one–dimensional resource (Min–Min and Max–Min).

In [80, 81], computing capacity was used as the main allocation criterion for VM creation and load minimization. Different algorithms, i.e., Greedy Worst–Fit, Sequential, Knapsack, and Time–bound Knapsack, were then presented. The experimental results using synthetic benchmarks show that Time–bound Knapsack achieves a 72% average load balancing, whereas the three other algorithms all result in loads above 80%. In [84], two algorithms called Dynamic Round Robin (DRR) and a combined DRR with First–Fit (Hybrid) were proposed, then compared with both the Round Robin and Power Save scheduling strategies in Eucalyptus. Simulation results showed that DDR and Hybrid decrease the power requirements by 56.5% and 55.9% compared with Round Robin, respectively. DDR and Hybrid also result in 3% less power consumption on average, compared with Power Save. A major limitation of these approaches is that, in practice, different VM requests may have different demands in terms of resource types. For instance, a request for general–purpose applications includes a balanced amount of CPU, memory, storage and network resources; a request for computer–intensive applications needs more CPU resources; and a request for data base and memory caching applications needs more memory resources. Therefore, solutions that consider only one resource as the main allocation criterion may be efficient for some resource dimensions while underutilizing others, thus resulting in higher actual costs.

Most of the existing work on VM placement in data centers fails to take advantage of the Min, Max, and Share parameters [23]. Particularly, the Min parameter ensures that VM receives the minimum amount of resources when powered on while the Max parameter ensures the maximum amount of resources for a VM to run a low–priority application. On the other hand, the Share parameter advises the Virtual Machine Monitor (VMM) to distribute resources among contending VMs. These parameters allow VMs to run heterogeneous applications, wherein the amount of resources allocated to a VM may be adjusted based on available resources, power costs, and application utilities. The same work [23] also introduced

a suite of algorithmic techniques to address the Min–, Max–, and Share–aware placement problem called GreedyMax, GreedyMinMax, and ExpandMinMax (respectively). The authors then proposed a VM placement algorithm called PowerExpandMinMax. Simulation results on a range of large synthetic data center setups and a small real data center testbed have shown that leveraging such parameters may improve the overall utility of the considered data centers by 47% or more. The limitation of this work is that it considered CPU utilization only and was limited to homogeneous servers.

Some works in the literature actually considered multiple resources for VM placement [26, 52, 75, 97, 125]. For instance, a vector–based scheme was employed by the HARMONY system for load balancing [125]. In such solutions, resources were normalized along dimensions as vectors, e.g., the resource requirement vector of VMs (RRV), the total resource capacity vector of servers (TCV), and the utilized capacity vector of servers (UCV). The VectorDot metric, for instance, was defined as the dot product of RRV and UCV and was proposed as a basis for choosing the target physical server for placing VM. Accordingly, after accepting the VM depending on the RRV, the placement scheme selects the physical server whose UCV gives the lowest value derived from the dot product. Evaluation results on simulated data center environments of various sizes and a real data center testbed have shown that VectorDot in combination with Best–Fit (BFVD), First–Fit (FFVD), Worst–Fit (WFVD), and RelaxedBestFit (RBFVD) achieves a more balanced resource usage compared to traditional greedy algorithms. For instance, the average system load obtained by using VectorDot is quite high (i.e., 70%). A different metric based on the vector–based approach, i.e., the *cosine of the angle* between UCV and TCV, was also introduced in [26] to rank physical servers. In detail, the authors proposed a scheme for balancing the utilization of multiple types of resources by minimizing the angle between UCV and TCV among all physical servers. Simulation results using randomly–generated data have shown that the cosine of the angle makes the VM deployments much more balanced in multiple resource utilization among the servers compared to the method used in Sandpiper [144] (see also the discussion below).

A market–based solution for multiple–resource load balancing was presented in [147] in the context of job scheduling. For such a scenario, a Market Mechanism (MM) scheme was introduced to balance multiple resources among servers with heterogeneous capacities. The authors pro-

posed a selection policy built on top of a pricing model to calculate the cost per resource of a job. The policy chooses the server whose utilization across different types of resources gives the lowest cost value. Simulation results using randomly–generated data with uniform distribution have shown that MM performs better than Latest Arrival Job (LAJ), Backfill Lowest (BL), and Backfill Balance (BB) in terms of load balancing. Furthermore, MM maintains a high server utilization (i.e., 80 – 85%). The work in [147] targets both homogeneous and heterogeneous physical servers and considers CPU, memory, and network bandwidth.

Sandpiper [144] combined three dimensions into a single *volume* metric as the product of its CPU, memory, and network utilization. The higher utilization of multiple resources, the higher value of the volume metric. The same work [144] also introduced a black–box and gray–box (BG) strategy that adopts the volume metric to select target physical servers. In particular, servers are sorted by ascending volume then considered for allocation of VMs under migration. The implementation of Sandpiper based on Xen has demonstrated that it may respond to network, CPU, or memory hotspots and may collocate VMs that stress different resources to improve overall system utilization (i.e., the aggregate CPU and network utilizations on both servers falls below 50%). Different VM placement algorithms have then been implemented to improve the resource utilization of each dimensions by taking the volume metric as the main optimization objective [97, 128, 135]. However, simple extensions of well–known VM placement solutions may cause low resource utilization and unbalance the load, thus requiring more active servers than those really needed and increasing operational costs.

2.3 System Model and Considered Metrics

The target system is an IaaS environment, i.e., a large–scale cloud data center consisting of M heterogeneous physical servers, denoted as $P = \langle p_1, p_2, \ldots, p_M \rangle$. Each server is characterized as $p = \langle pi,\, vm,\, \hat{r}^d \rangle$ with multiple types of resources, $d \in \{1, ..., D\}, D \in \mathbb{N}$, where: pi is the unique identifier of a server; vm is a set of VM instances that are allocated in p; and $\hat{r}^d = \{\hat{r}^1, \hat{r}^2, ..., \hat{r}^D\}$ describes the type and amount of the d–th resource consumed, where each dimension corresponds to a given type of physical resource (i.e., CPU, memory, storage, and network bandwidth).

Multiple independent cloud users submit requests for provisioning of N

heterogeneous VMs, denoted as $V = \langle v_1, v_2, \ldots, v_N \rangle$, which can be represented similarly to the multi–resource dimensions of physical servers. In detail, a VM is represented as $v = \langle vi, r^d \rangle$.

Let $U_t^d(p)$ be the utilization of resource $d \in \{1, ..., D\}$ of a server p at time t. Then, $U_t^d(p)$ of type d is defined as the total resource usage of all running VMs in p divided by the total resource capacity of the considered server:

$$U_t^d(p) = \frac{u_t^d(p) + w^d(p)}{\hat{r}^d(p)}, \tag{2.1}$$

where $u_t^d(p)$ is the total resource usage of the d–th dimension of an already–placed set vm of VMs that are allocated to p at time t, $u_t^d(p) = \sum_{v \in vm(p)} r^d(v)$; $w^d(p)$ is the initial load of the d–th resource of p.

Let $u_t(p) = \frac{1}{|D|} \sum_{d=1}^{|D|} U_t^d(p)$, $d \in \{1, \ldots, |D|\}$ be the mean value of the resource utilization across the $|D|$ dimensions, i.e., the overall average resource utilization of p at time t. Additional metrics are defined as follows.

Hottest resource ratio The hottest resource ratio $RH_t(p)$ of a server p is defined as the maximum value of the resource utilization

$$RH_t(p) = \max_{d \in \{1,\ldots,|D|\}} U_t^d(p). \tag{2.2}$$

The value of the $RH_t(p)$ metric is lower than or equal to one; the case $RH_t(p) = 1$ implies full load on at least one of the resources types in server p. In such a case, the most utilized resource is the critical resource that becomes a bottleneck for the server. We also define the type (or dimension) of the hottest resource $\hat{d} \in \{1, \ldots, |D|\}$ as:

$$\hat{d} = \operatorname*{argmax}_{d \in \{1,\ldots,|D|\}} U_t^d(p). \tag{2.3}$$

Resource balance ratio The resource balance ratio $B_t(p)$ of a server p is defined as the overall resource utilization divided by the hottest resource ratio

$$B_t(p) = \frac{u_t(p)}{RH_t(p)}. \tag{2.4}$$

Resource temperature ratio The resource temperature ratio $RT_t(p)$ of an overutilized server p is defined as the hottest resource ratio beyond the hot threshold

$$RT_t(p) = RH_t(p) - h^{\hat{d}}. \tag{2.5}$$

Resource correlation ratio The resource correlation ratio $RC_t(v, p)$ is defined as the absolute value of the resource of type $\hat{d}$ required by the

selected VM v, corresponding to the amount of the same resource type that needs to be allocated

$$RC_t(v,p) = |RT_t(p) \cdot \hat{r}^{\hat{d}}(p) - r^{\hat{d}}(v)|. \tag{2.6}$$

2.4 Virtual Machine Placement for Balanced Resource Utilization

The VM placement problem arises at the time cloud users submit a set of VMs to IaaS clouds. A related goal is to find the most suitable physical servers in a large cloud data center to accommodate the required VMs so as to minimize the number of used servers. In most scenarios, when cloud users send their VM requests, some physical server(s) in the cloud data center will be selected to deploy the required VMs. Most of the existing solutions only consider a limited number of resource types (e.g., a CPU resource only), thus resulting in unbalanced load of other resources. Such unbalanced use of resources may result in the unnecessary activation of physical servers. For instance, the Eucalyptus cloud management middleware [57, 68, 84] uses a simple greedy First–Fit algorithm for placing VM requests; it assigns a VM request to the first scanned physical server that satisfies the demands of all resources for that specific VM. OpenNebula [28, 107] uses a load–aware policy that selects the physical server with the most unused CPU first for allocating VM requests. The Min–Min [43, 53, 127] heuristic algorithm assigns the VM request with the lowest CPU capacity first to the server with the highest available CPU capacity. On the other hand, the Max–Min [43, 53, 127] heuristic algorithm assigns the VM request with the highest CPU capacity first to the server with the highest available CPU capacity.

Other approaches actually considered multiple resources by combining the usage of CPU capacity, memory, storage, and network into a single metric for VM placement. For instance, the VectorDot approach [125] assigns a VM request with the resource requirement vector (RRV) to the physical machine whose utilized capacity vector of servers (UCV) gives the lowest dot product value after accepting that VM. The Market Mechanism (MM) approach [147] assigns a VM to the server whose resource utilization gives the lowest cost for that specific VM. The algorithm proposed in [26] (referred to as Max–Cos in the following) assigns a VM request to the server with the lowest cosine value between the UCV and the total resource capacity vector of servers (TCV) of the physical machines.

In contrast, Publication I considers multiple types of resources as CPU capacity, memory, storage, and network bandwidth. Accordingly, Publication I proposes the Max–BRU algorithm that not only optimizes the resource utilization but also balances the usage of resources across multiple dimensions, so that the total number of active servers is minimized. The main idea of Max–BRU is the joint use of the resource utilization and the resource balance metrics across different types of computing resources. Therefore, a target physical server may accommodate many different types of VMs, thus requiring less active servers than those that would otherwise be needed. As discussed in Section 2.1, the low server utilization and an unbalanced use of resources across multiple dimensions may result in the unnecessary activation of physical servers, thus increasing actual costs. To this regard, the number of running servers is extremely important since it induces the majority of the operational costs. Publication I assumes that physical servers are initially empty (i.e., physical servers do not host any VMs) and the multiple types of resources are CPU capacity, memory, storage, and network bandwidth. The Max–BRU algorithm considers the VM resource requirements and the physical server resource capacities while placing VMs. In the following, the Max–BRU algorithm is introduced. Then, the key evaluation results are summarized.

2.4.1 The MAX–BRU Algorithm

Our research objective is to maximize the server utilization and balance the load across multiple types of resources. Consequently, both the $U_t^d(p)$ and $B_t(p)$ metrics are employed as the main allocation criteria. The most appropriate physical server for deploying VMs is determined based on them. Max–BRU is illustrated by Algorithm 1. In detail, the VM placement algorithm takes as input the submitted VMs in terms of their resource requirements and derives the number M of activated physical servers together with the average resource balance ($\hat{B}_t$) over all the powered–on servers at time t. Particularly, the VM placement procedure to compute VMs to physical servers allocation is as follows.

- The most appropriate server $p \in P$ with the lowest value of $RH_t(p)$ combined with the lowest $B_t(p)$ is chosen at each step in the set M of the currently activated servers (line 5).

Algorithm 1: Max–BRU(V)

Output: $M,\ \hat{B}_t$

1 Set $P \leftarrow \emptyset$, $M \leftarrow 0$;
2 **while** $V \neq \emptyset$ **do**
3 Set $status \leftarrow false$;
4 **if** $P \neq \emptyset$ **then**
5 $p \leftarrow \begin{cases} \min_{p \in P} RH_t(p); \\ \min_{p \in P} B_t(p); \end{cases}$
6 $v \leftarrow \begin{cases} \max_{v \in V} r^d(v); \\ r^d(v) + u_t^d(p) + w^d(p) \leq \hat{r}^d(p); d \in \{1, \ldots, |D|\}; \end{cases}$
7 **if** $v \neq \varnothing$ **then**
8 Place VM v on physical server p, update $U_t^d(p)$ and $B_t(p)$;
9 $V \leftarrow V \setminus \{v\}$; Set $status \leftarrow true$;
10 **if** $status = false$ **then**
11 Remove VM request v from V in FIFO order;
12 Start–up an inactive physical server p_{inact};
13 Place VM v on p_{inact}, compute $U_t^d(p_{inact})$ and $B_t(p_{inact})$;
14 $P \leftarrow P \cup \{p_{inact}\}$;
15 Increase M by 1;
16 $\hat{B}_t \leftarrow \frac{1}{M} \sum_{p \in P} B_t(p)$;
17 **return** $M,\ \hat{B}_t$

- The considered VM request v is the one with the highest $r^d(v)$ resource required, corresponding to the lowest d–dimensional resource utilization of a given server p, i.e., $\max_{v \in V} r^d(v)$, and satisfying the demands for all resources, i.e., such that $r^d(v) + u_t^d(p) + w^d(p) \leq \hat{r}^d(p);\ d \in \{1, \ldots, |D|\}$ (line 6).

- If V has a suitable VM request, that is then placed to the selected server p (lines 7–8). Otherwise, an inactive server p_{inact} is powered–on for allocating the first VM request v from V in FIFO order, then p_{inact} is added to the set P (lines 10–14). Note that, when v is allocated to p, v is removed from the set V and the server p is updated with new values for $U_t^d(p)$ and $B_t(p)$.

This algorithm effectively reduces the number of activated servers by

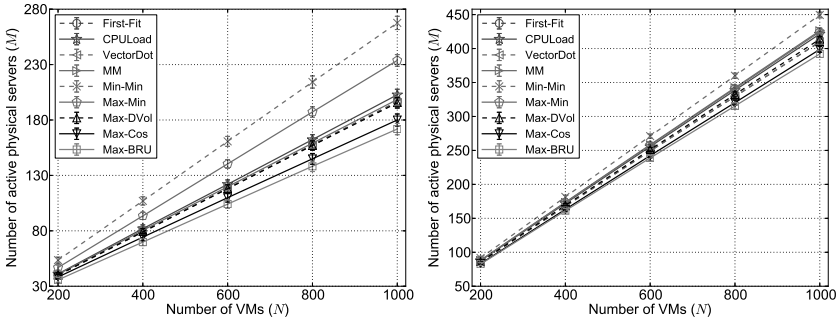


Figure 2.2. Number of active servers as a function of the number of VM requests (Publication I) for the (a) Amazon EC2 and (b) normal datasets.

improving server utilization, while balancing the usage of multiple types of resources. The Max–BRU algorithm is suited to VM placement due to its polynomial worst–case time complexity on the number of the submitted VMs (the details can be found in Publication I).

2.4.2 Summary of Results

The experimental results obtained by Max–BRU with respect to the existing solutions for VM placement are summarized next. They include a comparison with the state of the art discussed in Section 2.4.1 (i.e., First–Fit, CPULoad, VectorDot, MM, Min–Min, Max–Min, and Max–Cos) as well as an extension of the volume metric introduced in [144], referred to as Max–DVol in the following. Our evaluation focuses on the number of active servers (the lower the better) and the average resource balance ratio (the higher the better).

Number of Active Physical Servers

Figure 2.2 shows that existing solutions result in a higher number of running servers with respect to Max–BRU. Furthermore, the increase in the number of the physical servers activated by Max–BRU is slow when the VM requests increase. Specifically, in the case of resource requirements extracted from Amazon EC2 (Figure 2.2a), the number of active servers needed to allocate 1,000 VM requests is the smallest with Max–BRU, corresponding to 171.84 ± 5.11. In contrast, Min–Min, Max–Min, and CPULoad result in a higher number of servers used, because these algorithms take only CPU capacity as the main allocation criterion. As a result, some resources may be utilized more than others and a new server (i.e., an

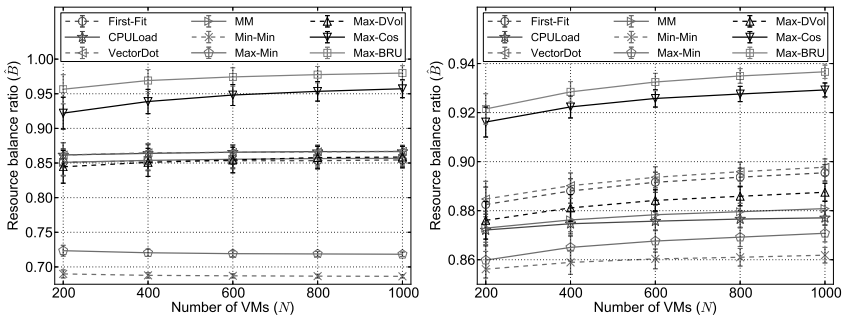


Figure 2.3. Resource balance ratio as a function of the number of VM requests (Publication I) for the: (a) Amazon EC2 and (b) normal datasets.

inactive server) may have to be powered–on for placing a newly–created VM, thus increasing costs. Max–BRU performs better than all other algorithms also when the resource requirements follow a normal distribution (i.e., Figure 2.2b), even though the gain is lower than in the EC2 dataset. This is due to the variability of the resources in the VM requests; in fact, the normal dataset demands the most active servers in general, followed by the EC2 dataset. Overall, the obtained results show that the proposed Max–BRU algorithm needs the smallest number of active servers in a cloud data center, even when the number of VM requests is high. Consequently, combining resource usage and load balancing across multiple resources results in a reduction of operational costs and energy expenditure (due to the smaller number of servers used); even a small improvement is considerable in large cloud data centers.

Resource Balance Ratio

Figure 2.3 illustrates the average resource balance ratio obtained for the different datasets. In this case, Max–BRU clearly obtains the best performance for both the EC2 (Figure 2.3a) and the normal (Figure 2.3b) datasets. The results show that Max–BRU is more sensitive to VM requests whose resource demands are highly varying. Nevertheless, Max–BRU is capable to spread the resource utilization uniformly over the different types of resources in all datasets. Similar to the previous results, Max–Cos is the only algorithm with comparable performance as it considers both UCV and TCV as metrics; Max–DVol also results in a lower resource balance ratio than Max–BRU, because it aims at maximizing the resource utilization rather than at balancing the load across multiple resources.

3. Efficient Virtual Machine Consolidation

In the previous chapter, I have presented a virtual machine (VM) placement algorithm for energy–efficient Infrastructure–as–a–Service (IaaS) cloud data centers. Experimental results have shown that Max–BRU limits the number of active servers while spreading the usage of resources across multiple dimensions. However, while providing a solution to the VM placement problem is important starting from the VM submission phase, VM consolidation algorithms are even more important to support continuous consolidation of already–placed VMs on the least number of physical servers. In such a scenario, VM consolidation executes periodically — for instance, every few minutes or hours, daily, weekly, or monthly — to reallocate already–placed VMs into as few physical servers as possible. The main contributions in this chapter are the implementation and evaluation of a VM consolidation solution that: (1) considers multiple types of resources, including CPU, memory, storage, and network bandwidth; (2) spreads the load across different types of resources while placing VMs; (3) supports an efficient method to estimate the future state of physical servers; and (4) provides an energy–efficient VM consolidation algorithm, which involves overloaded and underloaded host detection, live VM migration, VM placement under migrations, and idle server management.

This chapter presents the research done in Publication II, Publication III, and Publication IV. First, an overview of overloaded and underloaded host management as well as VM consolidation is presented. Second, a multiple resource selection (MRS) scheme and its algorithm for VM consolidation are introduced (Publication II). Then, an embedded multiple usage prediction (MUP) scheme for VM consolidation algorithm across multiple resources and a tunable prediction horizon called VMCUP–M are described (Publication III and Publication IV). The MUP scheme al-

lows a VM consolidation algorithm to scale the cloud data center power consumption according to the variability in the resource load (i.e., switching idle nodes to low–power states for energy saving purposes). Publication II and Publication IV target multi–resource types while Publication III and Publication IV address heterogeneous physical servers, heterogeneous VMs (i.e., VM instances with different sizes and resources), future state of servers, and idle server management. The MRS algorithm is evaluated through a custom simulator written in Java while the VMCUP–M algorithm is evaluated by extending the CloudSim simulation toolkit [22]. According to the experimental results from both synthetic and real–world workloads, the proposed algorithms significantly reduce the number of active servers, the number of migrations, and the number of power state changes, thus minimizing the power consumption of IaaS cloud data centers while complying with the service level agreement (SLA).

3.1 Overloaded and Underloaded Host Management

Determining whether a host is overloaded or underutilized then reallocating VMs from those hosts possibly affects the quality of service (QoS) experienced by hosted applications. In this context, several VM consolidation schemes have simply taken the current utilization of a single resource (e.g., CPU) into account while deciding whether a physical server is overloaded or underutilized [12, 80]. Other schemes consider the current CPU, memory, storage, and (or) network usage, then transform them into a single metric [125, 144, 145, 147]. In any case, as they are purely based on the last observed utilization for decision making, existing solutions may cause unnecessary migrations and eventually increase overheads: the energy for VM migration, the performance degradation of the hosted applications, and the extra network communication [37, 50, 133].

Initial studies in this context used static hot and cold thresholds to determine whether a host is overloaded or underutilized. As a consequence, these approaches keep the current (CPU) utilization of a server between the two thresholds. However, setting static thresholds and using the current utilization of a single resource are not effective measures for environments with dynamic workloads, in which the utilization of VMs running on a physical server continuously change over multiple resource dimensions. The authors in [12] proposed a set of metrics to rank physical servers by considering an adaptive upper bound based on a statistical

analysis of historical CPU data. Even though the used thresholds are not static, these approaches only leverage the current CPU utilization as the main criterion to decide on VM migrations. Thus, they do not allow for a reliable characterization of overloaded and underloaded servers, eventually resulting in unnecessary migrations and energy wastage. The impact of multiple types of resources on the detection of hot and cold servers was not considered in [12] either.

Some works in the literature actually considered multiple resources for overloaded and underloaded host management [12, 80, 125, 144, 145, 147]. In this context, several metrics have been proposed to rank physical servers. Among them, a VM management system was introduced in Sandpiper [144] to detect and resolve overload situations. In detail, Sandpiper used a single *volume* metric as a criterion to detect overloaded servers; a black–box and gray–box (BG) strategy considers the server with the highest volume.

In [45], the authors presented a resource pool management to detect and react to underload and overload situations. Particularly, a physical server is considered overloaded if the utilization of its CPU or memory are above a predefined threshold. Once an overloaded server is found, a fuzzy controller is used to select the VMs to be migrated as well as the candidate destination physical server to accommodate the migrated VMs. In such a case, the least loaded server that satisfies the demands for all the resources of the selected VM is selected as the target server. If there is no physical server with sufficient resources available, a new server is powered on and the VM is migrated to this newly–started server. On the other hand, the underutilized situation is identified when the average resource utilization of all active physical servers is below a given threshold. In such a case, the fuzzy controller chooses the least loaded server and attempts to move all its VMs to other suitable servers and switches the considered server to a power–saving state (e.g., suspend). Simulation results show that the best CPU and memory overload thresholds are 85% and 95%, while the best underload thresholds are 50% and 80%, respectively. The major limitation of the resource pool management is that it only considers CPU and memory resources as well as homogeneous physical machines.

3.2 Virtual Machine Consolidation

Virtual machine consolidation allows IaaS cloud data centers to reduce resource fragmentation or create idle physical servers during periods of high or low resource utilization, respectively. This may be achieved via repacking already–placed VMs on the least number of physical machines. This section overviews relevant approaches proposed in the literature for VM consolidation in large cloud data centers.

pMapper [139] targets CPU utilization by a power and migration cost–aware application placement framework for heterogeneous and virtualized computing systems. Particularly, pMapper consists of three main algorithms: min power parity (mPP), min power placement with history (mPPH), and balance between power and migration cost (pMaP). mPP is designed to minimize total power consumption only. It takes the VM sizes, the current assignment of VMs to physical machines, and the power model of all servers as input. mPP then sorts VMs and physical machines in decreasing order of their utilization, then places the VMs onto physical servers by using the First–Fit–Decreasing heuristic. mPPH is an extended version of mPP that reduces the migration cost by migrating as few VMs as possible. Finally, pMaP optimizes the power and migration cost trade–off while placing VMs. The authors also propose a variant of pMaP, namely pMaP+, which is particularly suitable for high loads. The major limitations of pMapper are that it only considers CPU and does not support intelligent overloaded and underloaded host management mechanisms.

Sercon [98] considers two dimensions (i.e., CPU and memory) for consolidating VMs. Particularly, Sercon modifies the First–Fit–Decreasing to limit the number of active physical servers while minimizing the number of VM migrations. Sercon sorts the list of physical servers based on their resource utilization, then considers the server with the least load first. The VMs from the least loaded server are sorted in decreasing order according to their resource utilization. This algorithm tries to migrate all VMs from underutilized servers to other suitable servers. Simulation results show that such an algorithm reduces the number of migrations compared to FFD while requiring only up to 6% more active physical servers. Sercon considers CPU and memory and was evaluated in a homogeneous environment. It does not support migrating underloaded servers and switching idle hosts to low–power states to save energy.

A black–box and gray–box (BG) strategy was introduced in Sandpiper, based on the *volume* criterion, for VM consolidation in large data centers [144]. BG sorts the list of overloaded servers based on their volume metric and the VMs in each server based on their volume–to–size ratio (VSR). It then considers the server with the highest volume first; the VM that needs to be migrated is then the one with the highest VSR. The BG scheme also adopts the volume metric to select target physical servers, i.e., they are sorted by ascending volume to allocate the VMs under migration. A prototype data center evaluation on a cluster of 16 servers that runs a total of 35 VMs shows that Sandpiper reduces the number of intervals spent in sustained overload by 61%. Furthermore, simulation performed on 50 physical servers (each with 3 VMs) shows that Sandpiper eliminates all hot spots about 73% of the time when there are 35 overloaded servers, while a suitable solution was found only 3% of the time with 40 overloaded servers. Additional experiments also suggest a CPU threshold of 75% to absorb the CPU overhead of migration while maximizing server utilization; a similar threshold is used for network utilization. The major limitations of the Sandpiper system are that it considers homogeneous physical servers and does not support migrating underloaded machines.

3.3 Virtual Machine Consolidation with Multiple Usage Prediction

In contrast to the VM placement problem discussed in Chapter 2, which does not manage already–placed VMs, VM consolidation requires to migrate existing VMs live. Particularly, VM consolidation with live migration is closely related to the problem of: (1) determining when a server is overloaded (i.e., a *hot spot*), then migrating the potential VMs from such a server to maintain a certain QoS; and (2) determining when a server is underloaded (i.e., a *cold spot*), then migrating all VMs from such a server and switching a considered server to a low–power state. Both sub–problems above help remove resource fragmentation of servers and create idle servers to minimize energy consumption. However, existing VM consolidation algorithms may result in a high number of VM migrations and server switches, thus increasing the overheads: the energy for VM migration, the performance of applications hosted on the VMs, and the extra network communications. Therefore, this dissertation adds two other objectives to the VM consolidation problem: minimizing the number of VM

migrations and the number of server switches, along with the number of active physical machines. To enable VM consolidation with live migration and address the above–mentioned issues, Publication II, Publication III, and Publication IV make the following three contributions.

- VM consolidation takes the Max–BRU VM placement algorithm in Chapter 2 (Publication I) into account at the initial layout, thus helping to minimize the number of active servers; this further limits the number of VM migrations for reaching the consolidated stage.

- A multi–resource selection (MRS) scheme is proposed for consolidating VMs (Publication II). It characterizes multiple types of utilized resources, then ranks physical servers through the combined resource usage and load balancing across multiple dimensions. This helps remove resource fragmentation of servers and further increases the resource utilization of a data center. With MRS, the current hottest resource is used as the main criterion to determine when a server is considered overloaded or underutilized.

- An efficient multiple usage prediction (MUP) approach is presented to estimate the long–term utilization over a future time period of multiple resources types based on the local history of the considered servers (Publication IV). MUP is an extended version of a usage prediction (UP) approach proposed in Publication III. Furthermore, Publication II is limited to the current utilization of multiple types of resources while Publication III and Publication IV use both the current and predicted usage metrics to characterize and classify a physical server, thanks to MUP. This allows for a reliable characterization of overloaded and underloaded servers, thus limiting the number of unnecessary VM migrations and server switches with the goal to reduce the energy consumption of a cloud data center.

This section begins by discussing the multi–resource selection (MRS) scheme for consolidating VMs in Publication II. Then, multiple usage prediction (MUP) and the performance of MUP are presented (Publication IV). After that, overloaded and underloaded host detection, VM selection and placement, and the VMCUP–M algorithms are detailed (Publication IV). Finally, the main evaluation results of the three publications are

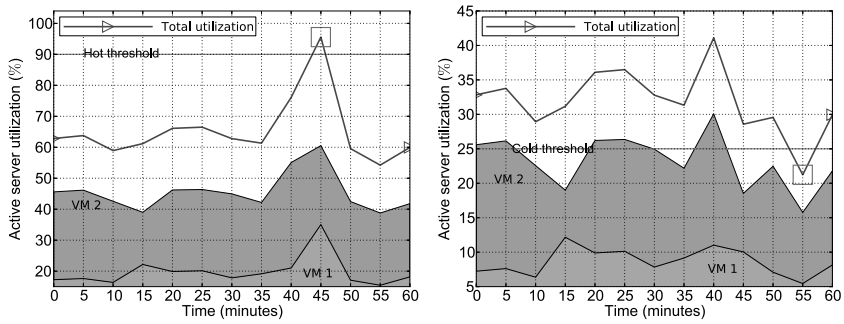


Figure 3.1. Host management with MRS in Publication II: (a) overloaded server and (b) underloaded server detection.

summarized. The full experimental results are detailed in the publications attached at the end of the dissertation.

3.3.1 Multiple Resource Selection

The working of the multiple resource selection (MRS) scheme is illustrated in Algorithm 2. The goal is to reallocate VMs from overloaded and underloaded servers into as few physical machines as possible, thus improving resource utilization and saving energy. To achieve this, MRS considers the VM consolidation problem in a scenario where no historical resource utilization data is used. In particular, VMs are consolidated according to the current hottest utilization of resources applying the hot and cold spots detection and VM placement under migrations. In detail, a server is considered as overloaded if its hottest resource is above a hot threshold $h^{\hat{d}}$ for the resource type $\hat{d}$ (Figure 3.1a). Similarly, a sever is considered underloaded if its hottest resource is below a cold threshold $c^{\hat{d}}$ (Figure 3.1b). In such scenarios, the static hot threshold was set to $h^{\hat{d}} = 0.9$ and the cold threshold to $c^{\hat{d}} = 0.25$ for all $\hat{d} \in \{1, \ldots, |D|\}$ respectively, according to [145]. In the following, the hot and cold migration procedures are detailed.

- The hottest server h is handled first, then all VMs running on h are sorted in ascending order of their RC (lines 7–8). The VM v_i with the lowest RC is selected for migration (line 9).

- The appropriate server for placing the migrating VM v is obtained by the FINDPOTENTIALSERVER function (details can be found in Algorithm 2

Algorithm 2: MRS(P)

Output: $M,\ \hat{U}_t$

Set $M \leftarrow P.size$, $H \leftarrow \emptyset$, $C \leftarrow \emptyset$, $W \leftarrow \emptyset$;
for $\forall p_i$ **in** P **do**
    if $RH_t(p_i) > h^{\hat{d}}$ **then** $H \leftarrow H \cup \{p_i\}$;
    if $RH_t(p_i) < c^{\hat{d}}$ **then** $C \leftarrow C \cup \{p_i\}$;
Sort all servers h **in** H in descending $RT_t(h)$;
while $H \neq \emptyset$ **do**
    Remove h from H in $RT_t(h)$ order;
    Sort all VM v in h in ascending $RC_t(v, h)$;
    for $\forall v_i$ **in** h **do**
        $p \leftarrow$ FINDPOTENTIALSERVER($P \setminus \{H \cup C\}, v_i$);
        if $p = \varnothing$ **then** $p \leftarrow$ FINDPOTENTIALSERVER(C, v_i);
        if $p \neq \varnothing$ **then**
            Place v_i on p, update $U_t^d(p)$ and $B_t(p)$;
            if $p \in C, RH_t(p) \geq c^{\hat{d}}$ **then**
                $C \leftarrow C \setminus \{p\}$;
            break;
Sort all servers c in C in ascending the placed VM sizes;
while $C \neq \emptyset$ **do**
    Set $status \leftarrow true$;
    Remove c from C in placed VM sizes order;
    for $\forall v_i$ *in* c **do**
        $p \leftarrow$ FINDPOTENTIALSERVER($P \setminus C, v_i$);
        if $p = \varnothing$ **then** $p \leftarrow$ FINDPOTENTIALSERVER($C \setminus \{c\}, v_i$);
        if $p \neq \varnothing$ **then** $W \leftarrow W \cup \{p\}$;
        else Set $status \leftarrow false$; **break**;
    if $status = true$ **then**
        for $\forall v_i$ **in** c **do**
            Remove server p from W in FIFO order;
            Place v_i on p, update $U_t^d(p)$ and $B_t(p)$;
            if $p \in C, RH_t(p) \geq c^{\hat{d}}$ **then** $C \leftarrow C \setminus \{p\}$;
        Switch c to a low–power mode, decrease M by 1;
$\hat{U}_t = \frac{1}{M} \sum_{p \in P} RH_t(p)$;
return $M,\ \hat{U}_t$

in Publication II). Specifically, such a function checks whether server p with the lowest $RH_t(p)$ and $B_t(p)$ among the M active servers is available to allocate v or not. If server p does not exist, another server is selected from the set of cold servers C (line 11). If there is no server with enough available resources, the next VM in h is considered.

- The cold server c with the lowest placed VM size is considered first (line 20). All VMs placed in c need to be migrated before switching c to a low–power mode. For each cold server c in C, if a set of potential servers W is found, then all placed VMs in c are migrated in sequence to a physical server p in W (lines 21–29). If at least one VM in c could not find the new placement, the underloaded server migration does not migrate the VMs and c is kept active.

3.3.2 Multiple Usage Prediction

The algorithm described in Section 3.3.1 has simply taken the current hottest utilization of resources into account while deciding when a physical server is considered overloaded or underutilized. However, a solution based on the last observed utilization for decision making may cause inaccurate hot and cold server detection. This may happen because of a temporary (or abnormal) increase or decrease in the VM workloads with time. Consequently, it is essential to design an efficient scheme to correctly take decisions on hot and cold spots. In a scenario where historical resource utilization data are available (i.e., the $n = 12$ most recent utilization values for each resource type d), Publication III first proposes an efficient usage prediction (UP) approach to estimate the short–term future CPU utilization based on this historical resource usage data of the considered servers (please refer to Publication III for the details about the algorithm itself, a complexity analysis, and a performance evaluation). Based on UP, the multiple usage prediction (MUP) scheme is introduced to predict the long–term usage of multiple resource types $d \in \{1, ..., D\}$ of a server p over a time period $K \in \mathbb{N}^+$. This requires predicting the usage of the resource type d at the $k \in \{1, \ldots, K\}$ steps ahead in time, i.e., $U^d_{t+k}(p)$, from a current resource utilization $U^d_t(p)$. Specifically, multiple usage refers to both the multiple–resource and multiple–step usage, i.e., $U^d_{t+1}(p)$, $U^d_{t+2}(p)$, ..., $U^d_{t+k}(p)$. Therefore, MUP is achieved by iterating the usage prediction (UP) scheme detailed in Algorithm 3, corresponding to

Algorithm 3: UP(p, d, m)

Set $\mathbf{X} \leftarrow \mathbf{0}$, $\mathbf{y} \leftarrow \mathbf{0}$, $\boldsymbol{\beta} \leftarrow \mathbf{0}$;
// Training dataset: $\mathbf{X}$ (input) and $\mathbf{y}$ (output)
for $t = 0$ *to* $n - m$ **do**
    $\mathrm{X}_{t,0} = 1$;
    for $i = 0$ *to* m **do**
        $\mathrm{X}_{t,i+1} = U_i^d(p)$;
    $y_t = U_t^d(p)$; //$y_t \in \mathbf{y}$
// Estimate the regression coefficients $\boldsymbol{\beta}$ with OLS
$\boldsymbol{\beta} \leftarrow (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$;
// Estimate the future resource usage
$U_{t+1}^d(p) = \mathbf{U}^d(p) \cdot \boldsymbol{\beta}$;
return $U_{t+1}^d(p)$.

Prediction step	Inputs of MUP	Output	Usage prediction
1	$U_t^d(p)$	$U_{t+1}^d(p)$	UP$(p, d, 1)$
2	$U_t^d(p), U_{t+1}^d(p)$	$U_{t+2}^d(p)$	UP$(p, d, 2)$
3	$U_t^d(p), U_{t+1}^d(p), U_{t+2}^d(p)$	$U_{t+3}^d(p)$	UP$(p, d, 3)$

Table 3.1. The multiple–resource and multiple–step usage prediction ($m = 1$, $d \in D$ and $K = 3$) in Publication IV.

the regressor size $(m + k - 1)$. The details of MUP by iterating UP are illustrated in Table 3.1.

To better focus on the performance of the proposed MUP approach used, Figures 3.2 and 3.3 show the values predicted by MUP for CPU and memory compared to the real resource usage. The results show that the values predicted by MUP are always close to the real ones even during peaks. Specifically, the MUP prediction scheme obtains the best performance for a one–step CPU and memory usage prediction (Figures 3.2a and 3.3a), while it gets worse as the number of steps increases, i.e., when the number of steps is equal to six (Figures 3.2b and 3.3b). The underlying reason for this behavior is the distribution of the actual resource utilization, which is linear in time.

3.3.3 Overloaded and Underloaded Host Detection

Publication III and Publication IV improve Publication II by jointly using the current and predicted utilization metrics over multiple types of resources. This improvement allows for a reliable characterization of over-

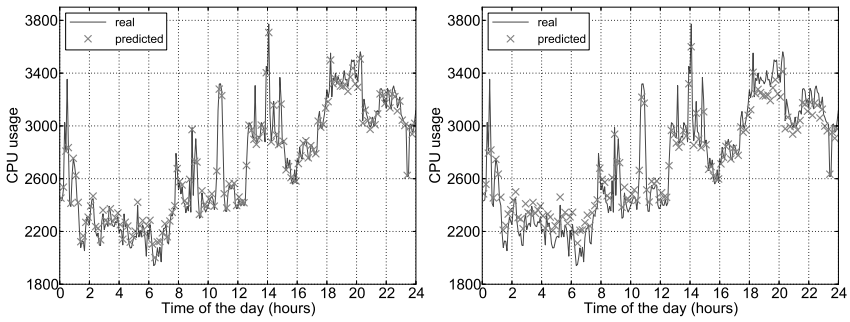


Figure 3.2. Prediction of CPU resource usage in Google Cluster Data (Publication IV): (a) one–step prediction and (b) six–step prediction.

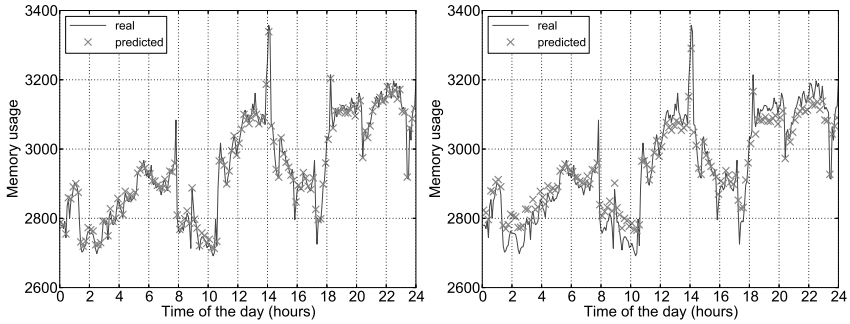


Figure 3.3. Prediction of memory resource usage in Google Cluster Data (Publication IV): (a) one–step prediction and (b) six–step prediction.

loaded or underloaded servers.

Each host periodically executes an overloaded host detection with multiple usage prediction (OHD–MUP) algorithm (Algorithm 4) to consolidate VMs when needed. The algorithm with MUP helps avoid the false hot detection of servers due to a temporary or abnormal increase in VM workloads with time compared to the scenario addressed in Publication II, where the current hottest usage values are taken. Accordingly, a server is considered *overloaded* (as shown in Figure 3.4a) if the following conditions are satisfied:

- the server is overloaded in both the current and the future period of time (top part of the Figure 3.4a), i.e., the current (the one marked with the rectangle) and the multiple predicted values (the small circles) of any resources are above a *hot threshold*;

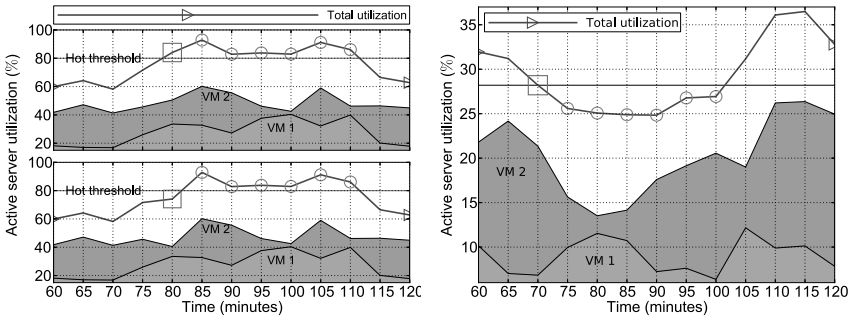


Figure 3.4. Host management with MUP (Publication IV): (a) overloaded server and (b) underloaded server detection.

Algorithm 4: OHD–MUP(p, D, m, K)

```
for ∀d in D do
    for ∀k in K do
        U^d_{t+k}(p) ← UP(p, d, m + k − 1);
        if U^d_{t+k}(p) ≤ h^d then return false;
return true
```

- the server is currently in a normal state but is overloaded in the future period of time (bottom part of the Figure 3.4a), i.e., the current utilization is below a *hot threshold* while the multiple predicted values are above a *hot threshold*.

Both conditions above indicate that the server is a potential candidate for migration because: (1) it is completely overloaded in both the current and the future period of time; or (2) it becomes overloaded in the near future. The *hot thresholds* used in Publication IV are static threshold (THR), i.e., the hot CPU and memory threshold is set to 80% of load, and a dynamic threshold based on local regression (LR), i.e., the hot CPU and memory threshold is estimated by using local regression of historical data.

To better understand how the joint use of current and future (i.e., predicted) resource usage metrics affect overloaded host detection, Figure 3.5 shows the CPU utilization measured every five minutes of a cloud server in our university over a 24 hour period of time. The results show that existing algorithms (e.g., those considering only the CPU resource) and the proposed algorithm in Publication II (i.e., based on the current hottest

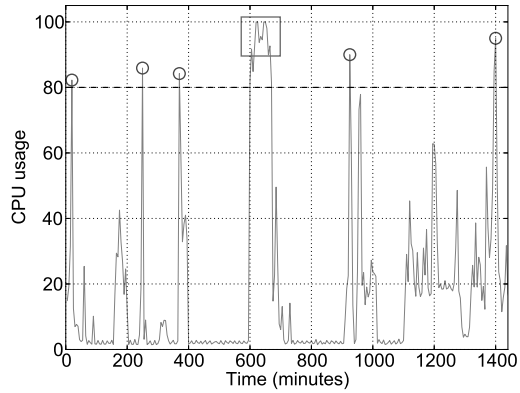


Figure 3.5. CPU resource usage measured every five minutes over 24 hours of a cloud server in our university (Publication IV).

resource utilization) as the main criterion to detect an overloaded server may cause an unreliable overloaded host detection. Specifically, the following criterion has been considered: a host is considered overloaded when the current CPU usage exceeds the threshold $h = 0.8$ (i.e., 80% of load). In Figure 3.5, the small circles on the top of the trace denote the false hot detection points, because the load of the considered host will rapidly decrease in the short–term future. In such a situation, the VMs allocated in a server do not need to be migrated to reduce the resource load. For the same trace, Publication III and Publication IV embedded the MUP scheme into the overloaded host detection algorithm (OHD–MUP) and reported only one point as a hot spot, the one marked with the rectangle in the period of time from 600 to 670 minutes. In this period, some VMs need to be migrated to avoid SLA violations. The example shows how MUP plays an important role in decision making for overload situations and how OHD–MUP avoids unnecessary VM migrations due to varying resource demands. In a cloud data center with millions of machines and a high variability of the VM workloads with time, a threshold–based current utilization may result in hundreds of hot spots or even worse. This has motivated us to develop an efficient method for a reliable characterization of overloaded servers. In fact, the proposed OHD–MUP leverages the current and multiple predicted utilization so as to limit the number of hot spots to the few ones that are really necessary.

Publication II defines a sever as underloaded if its hottest resource is below a static cold threshold $c^{\hat{d}} = 0.25$. However, setting static cold threshold and using the current hottest utilization are not effective means to detect

Algorithm 5: UHD–MUP(p, D, m, K)

```
for ∀d in D do
    for ∀k in K do
        U^d_{t+k}(p) ← UP(p, d, m + k − 1);
        if U^d_{t+k}(p) > U^d_t(p) then return false;
return true
```

underloaded servers. This is especially critical for IaaS cloud environments with dynamic workloads, in which the utilization of VMs running on a physical server continuously changes over multiple resource dimensions. This has motivated us to propose an efficient underloaded host detection with multiple usage prediction algorithm namely UHD–MUP (Algorithm 5) for a reliable characterization of cold spots. In detail, when no host is overutilized, the host p with the lowest value of maximum resource utilization is considered. A server is defined as *underutilized* if its multiple predicted resource is equal or below the current resource utilization (Figure 3.4b). This indicates that the server is currently underutilized and its load will decrease in the considered time period, thus the host is a potential candidate to be switched to a low–power mode to save energy. The joint use of both the current and multi–predicted utilization metrics allows host management to correctly identify underutilized servers. If the load of the considered server will increase above the current usage in any time instant during the considered period, the algorithm takes no action. Finally, after migrating all VMs on the underloaded server, the idle server is switched to a low–power mode for energy efficiency reasons; otherwise, the host is kept active.

3.3.4 VM Selection and Placement under Migration

Once an overloaded server p is considered, the next step is to select a potential VM running on p for migration to reduce the resource load. For such a case, the authors in [12] proposed a set of VM selection policies to select the suitable VMs to be migrated. In detail, the minimum migration time (MMT) policy selects a VM with the shortest time to complete a migration with respect to other VMs allocated to the host. The maximum correlation (MC) policy selects a VM with the highest correlation of the CPU utilization with the other VMs. The maximum utilization (MU) policy selects a VM with the highest CPU utilization. The random selection

Algorithm 6: PABFD–MUP(P, v, D, m, K)

```
1  Set p ← ∅; min ← MAX;
2  for ∀p_i in P do
3      if ∀d ∈ D; r^d(v) + u_t^d(p_i) + w^d(p_i) ≤ r̂^d(p_i) then
4          oldPower ← getPower(p_i);
5          Place v on p_i, update U_t^d(p_i);
6          newPower ← getPower(p_i);
7          incPower ← newPower − oldPower;
8          if incPower < min AND OHD–MUP(p_i, D, m, K) = false then
9              min ← incPower; p ← p_i;
10         else Release v from p_i, update U_t^d(p_i);
11 return p
```

(RS) policy randomly selects a VM according to a uniformly distributed.

Publication IV introduces a VM selection policy — namely, the minimum resource temperature (MRT) — that migrates a VM v to reduce the resource temperature of an overloaded server p the most. As migration is expensive, our research goal is to select only the VMs that contribute more to the host load. Let $vm(p)$ be a set of VMs currently allocated to the host p. The MRT policy finds a VM $v \in vm(p)$ such that $\forall a \in vm(p)$ the following condition holds:

$$\frac{RT(p|vm(p) \setminus v)}{RAM_u(v)} \geq \frac{RT(p|vm(p) \setminus a)}{RAM_u(a)}. \tag{3.1}$$

As the result of VM migrations, the target physical servers may become overloaded during periods of high resource utilization. Consequently, overload management is required to detect overload situations and decide which physical machines should be selected to accommodate the migrated VMs. Publication IV extends the power–aware best fit decreasing (PABFD) algorithm introduced in [12] for multiple–resource VM placement. In particular, the MUP scheme is embedded into PABFD through a new VM placement algorithm called PABFD–MUP (Algorithm 6). Accordingly, PABFD–MUP selects a target physical server not only based on the least increased power consumption but also based on its utilization stability, which is predicted by using the MUP scheme. Importantly, PABFD–MUP decreases the chance of the target host being overloaded in the future period of time after placing the migrating VM.

Algorithm 7: VMCUP–M(P, D, m, K)

```
1  //Overloaded server migration with MUP;
2  for ∀p_i in P do
3      while OHD–MUP(p_i, D, m, K) = true do
4          v ← MRT(p_i);
5          p ← PABFD–MUP(P \ {p_i}, v, D, m, K);
6          if p ≠ ∅ then
7              Place v on p, update U_t^d(p);
8          else if ∃p_inact then
9              Switch p_inact to an idle mode;
10             Place v on p_inact, update U_t^d(p_inact);
11             P ← P ∪ {p_inact};
12         else break;
13 //Underloaded server migration with MUP;
14 Set status ← true;
15 while status = true do
16     Set p ← p_0;
17     for ∀p_i in P do
18         if max_{d∈D} U_t^d(p) > max_{d∈D} U_t^d(p_i) then
19             p ← p_i;
20     if UHD–MUP(p, D, m, K) = true then
21         Set W ← ∅;
22         for ∀v_i in p do
23             s ← PABFD–MUP(P \ {p}, v_i, D, m, K);
24             if s = ∅ then
25                 status ← false;
26                 break;
27             else W ← W ∪ {s};
28         if status = true then
29             for ∀v_i in p do
30                 Remove server s from W in FIFO order;
31                 Place v_i on s, update U_t^d(s);
32             Switch p to a low–power mode;
33             P ← P \ {p};
```

3.3.5 The VMCUP–M Algorithm

Publication IV also proposes a VM consolidation with multiple usage prediction (VMCUP–M) algorithm (Algorithm 7). The key idea behind the VMCUP–M algorithm is to leverage both multiple–resource and multiple–step prediction for underloaded and overloaded host management, VM selection, and VM placement under migration, with the ultimate goal of reducing the energy consumption of a cloud data center. In particular, VMCUP–M executes periodically (i.e., every five minutes) to evaluate the VM consolidation process based on the current and future prediction about the resource usage of the considered servers. The detailed overloaded and underloaded server migration procedures are as follows:

- If a sever p_i is overloaded (line 3), the VM v allocated in p_i is selected for migration by the MRT policy in Eq. (3.1). The appropriate server for placing the migrating VM v is obtained by PABFD–MUP (Algorithm 6). If such a server p does not exist, an inactive server p_{inact} is switched to idle state for allocating the selected VM (lines 8–11). The VM placement should be rejected if a server p does not satisfy the demands of all the resources in v or is overutilized after accepting v in the current and future period of time.

- If a server p is underutilized, all VMs placed in p need to be migrated before switching p to a low–power mode. For each VM v_i in p, if a set of potential servers W is found by the PABFD–MUP algorithm (Algorithm 6), then all placed VMs in p are migrated in sequence to a physical server s in W (lines 22–31). To this end, the cold server p is switched to a low–power state. If at least one VM in a cold spot p could not find a new placement, the underloaded server migration procedure does not migrate the VMs and p is kept active. Note that the underloaded server migration procedure continues until a server with the lowest utilization has not been considered as cold spot.

This algorithm effectively reduces the energy consumption while limiting the number of migrations and power state changes along with the number of active physical servers. For the details about the time complexity analysis of VMCUP–M and the experimental setup, please refer to Publication IV.

3.3.6 Summary of Results

The following discussion summarizes the results obtained by our proposed schemes for VM consolidation. After studying the impact of multiple usage prediction, the achieved performance is characterized in terms of number of activated servers, resource utilization, energy consumption, and SLA compliance (i.e., the average number of SLA violations). In detail, SLA violations are considered due to both overutilization (i.e., the percentage of time during which active servers have experienced the 100% utilization of any resources) and migration (i.e., the overall performance degradation while migrating VMs) as defined in [12].

Impact of Multiple Usage Prediction

The data center size is a parameter that allows to clearly see the impact of MUP on the average number of hot and cold spots (Figure 3.6) as well as on the average number of active machines per data center (Figure 3.7).

As for the hot and cold spots, overloaded and underloaded host detection with MUP significantly reduce the number of real hot and cold spots in the system. Specifically, for the GCD workload trace (Figure 3.6a), MUP obtains a 94% (THR) and 87% (LR) factor reduction of both hot and cold spots compared to the algorithms without prediction for a cloud data center with 1,400 of machines. Additionally, for the PlanetLab workload trace (Figure 3.6b), MUP reduces the number of hot and cold spots of more than 87% for the considered thresholds. As a consequence, the algorithms correctly identify hot and cold servers in the system when using MUP.

As for the average number of active machines per data center, the obtained results show that the overloaded and underloaded host detection approaches with MUP need the smallest number of active servers, even when the size of data center is high (Figure 3.7). In contrast, the THR and LR algorithms without MUP result in the highest number of servers used. In fact, these algorithms take only the current resource utilization as the main detection criterion, thus, they fail to reliably detect hot and cold spots due to the varying resource demands of VMs. As a result, they incur in unnecessary migrations, eventually resulting in powering on additional servers.

Number of Active Physical Servers

Figure 3.8 compares the performance of MRS with the BG–DVol, VectorDot, and MM schemes for the different workloads. In this case, MRS

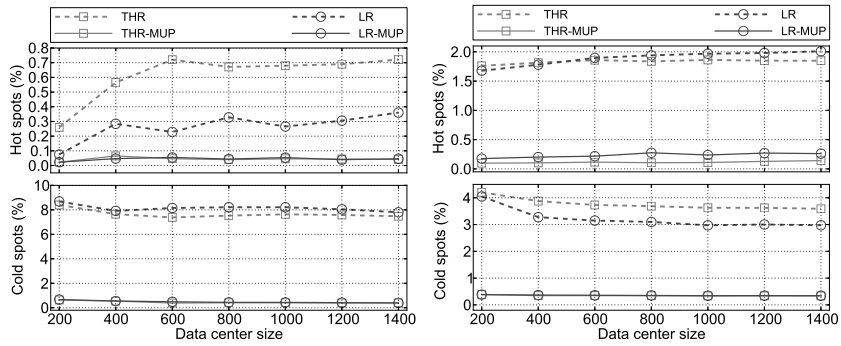


Figure 3.6. Impact of MUP on the average number of hot and cold spots per data center (Publication IV) for the: (a) GCD and (b) PlanetLab workloads.

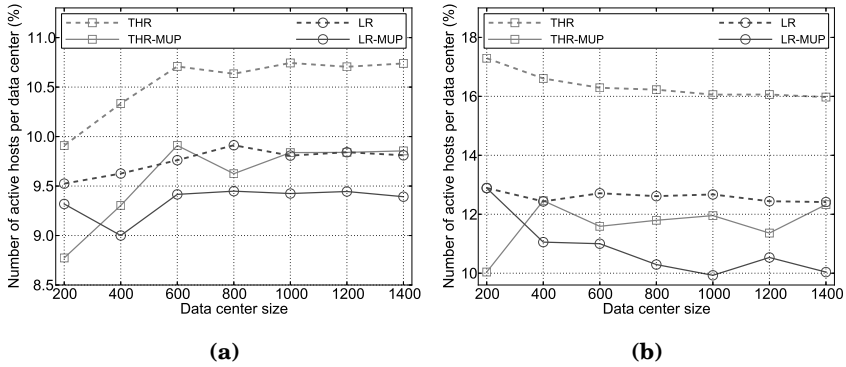
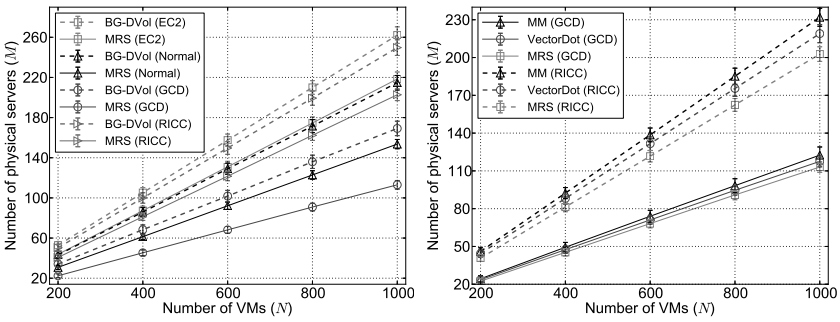


Figure 3.7. Impact of MUP on the average number of active machines per data center (Publication IV) for the: (a) GCD and (b) PlanetLab workloads.

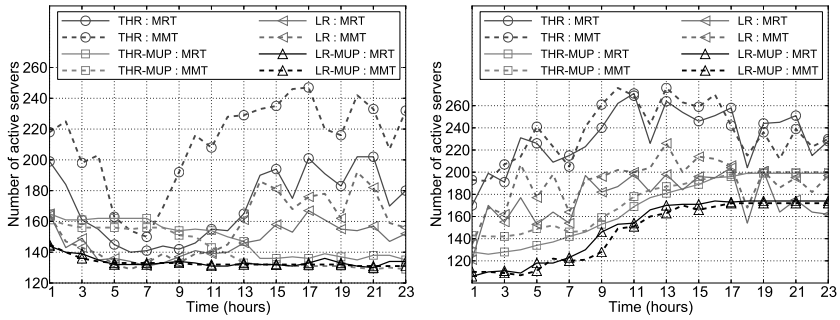
significantly reduces the number of active servers compared with BG–DVol (Figure 3.8a). Furthermore, MRS also obtains the best performance compared with two VM placement schemes, namely, VectorDot and MM, to find the best target physical machine and accommodate the VMs being migrated (Figure 3.8b). Overall, the obtained results show that the proposed MRS scheme needs the smallest number of active servers, even when the number of VM requests is high.

Figure 3.9 compares the performance of MRT with the minimum migration time (MMT) introduced in [10], with two different strategies: one with MUP and the other without MUP. In this case, MRT clearly obtains the best performance for both the THR and LR overutilized host detection approaches. Without MUP, the algorithms simply use the current



(a) **(b)**

Figure 3.8. Number of active physical servers as a function of the number of VM requests (Publication II) for the different workloads compared with: (a) BG–DVol and (b) VectorDot and MM.



(a) **(b)**

Figure 3.9. Number of active servers for MRT and MMT as a function of time (Publication IV) for the: (a) GCD and (b) PlanetLab workload traces.

resource load for making decisions. Therefore, they may incorrectly forecast overloaded and underloaded servers, thus resulting in unnecessary migrations and eventually increasing the number of active physical machines. When MUP is used, the VM consolidation procedure decreases the number of active servers to a small number that is almost constant over time. This is because VMCUP-M not only takes into account the current state of resources but also future usage. As a consequence, MUP helps avoid rapid changes in the number of active machines in a cloud data center.

Figure 3.10 compares the performance of the VMCUP–M and BG schemes (with and without MUP) in terms of the number of active hosts over time under the random (Figure 3.10a) and the GCD (Figure 3.10b) workloads. As a result, VM consolidation with MUP plays an important role

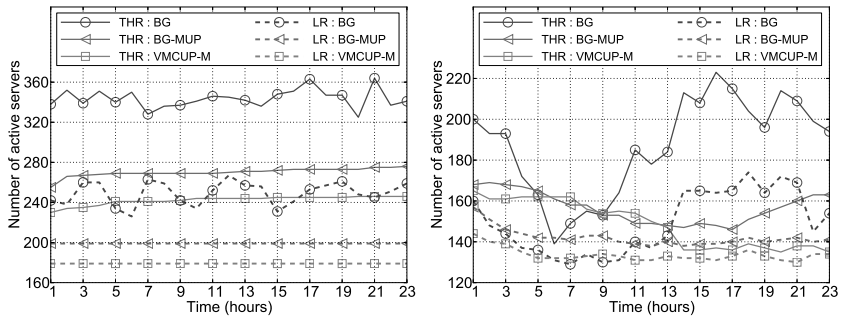


Figure 3.10. Number of active servers for VMCUP–M and BG as a function of time (Publication IV) for the: (a) random and (b) GCD workload traces.

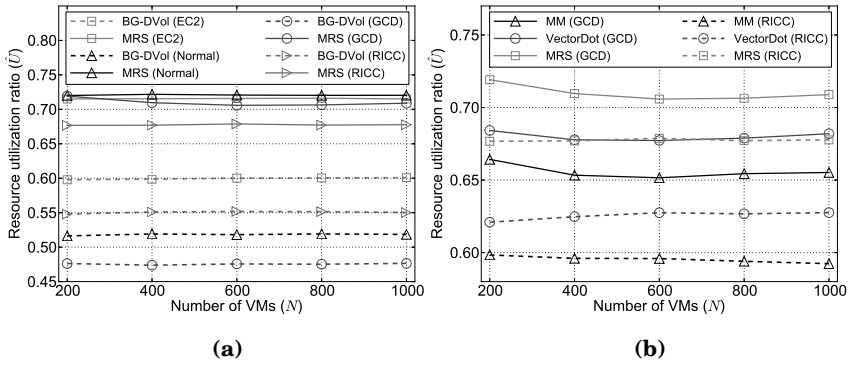
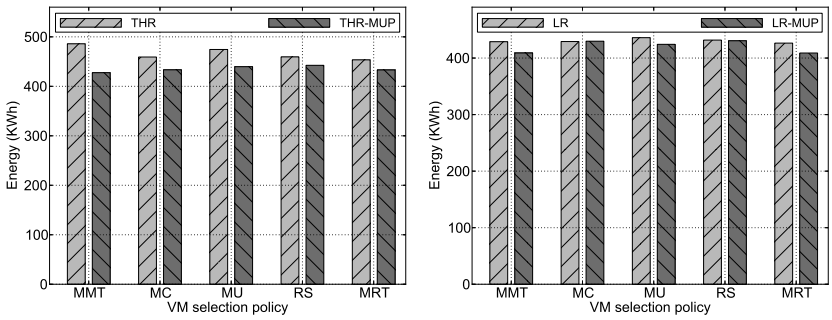


Figure 3.11. Resource utilization ratio as a function of the number of VM requests (Publication II) for the different workloads compared with: (a) BG–DVol and (b) VectorDot and MM.

in reducing the number of active servers in a data center. More importantly, VMCUP–M keeps the number of servers mostly constant during the simulation time.

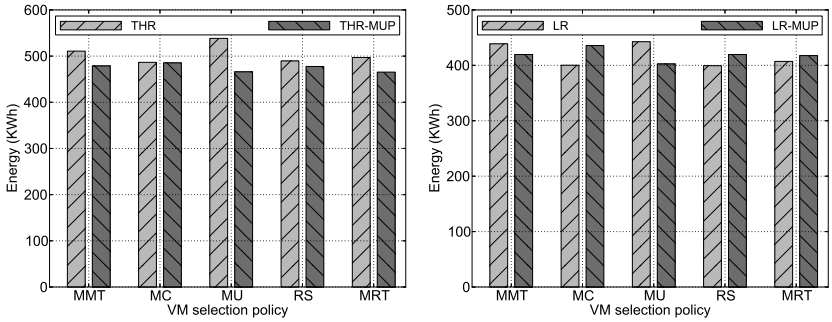
Resource Utilization Ratio

Figure 3.11 shows that the proposed MRS scheme achieves an improvement in the average resource utilization of about 12% (EC2), 21% (normal), 23% (GCD) and 12% (RICC), respectively. The improvement over BG–DVol, shown in Figure 3.11a, is more apparent for all considered workloads. MRS performs better than VectorDot and MM schemes over the real–world workloads from GCD and RICC (i.e., Figure 3.11b); the second–best selection scheme is VectorDot while the MM scheme performs much worse.



(a) **(b)**

Figure 3.12. Energy consumption of VMCUP–M for the GCD workload trace (Publication IV) with the: (a) THR and (b) LR overloaded host detection schemes.



(a) **(b)**

Figure 3.13. Energy consumption of VMCUP–M for the PlanetLab workload trace (Publication IV) with the: (a) THR and (b) LR overloaded host detection schemes.

Energy Consumption

The energy consumption of VMCUP–M under the different VM selection policies for the GCD workload trace with THR and LR is reported in Figures 3.12a and 3.12b, respectively. In this case, VMCUP–M reduces the power consumption of 5.6% or more (THR) and 4.6% or more (LR) with respect to algorithms without MUP. This indicates that VMCUP–M is able to correctly predict underutilized servers and to minimize the energy costs by switching idle hosts to a low–power state. Moreover, MRT also achieves the lowest energy consumption compared to all other VM selection policies even without prediction. Overall, the obtained results show that VMCUP–M combined with MRT significantly reduces the energy consumption by avoiding unnecessary migrations and server switches while

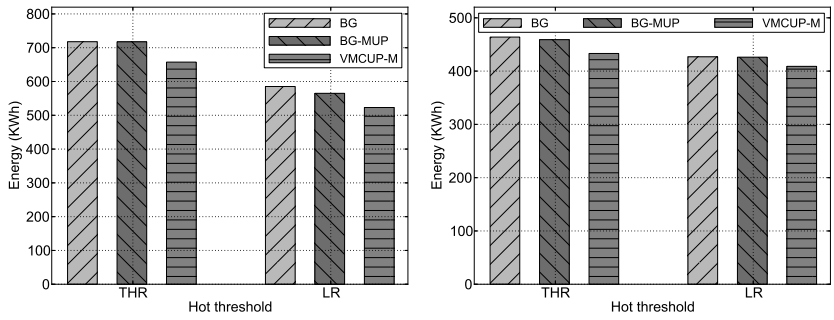


Figure 3.14. Energy consumption of VMCUP–M and BG under the THR and LR overutilized host detection approaches (Publication IV) with the: (a) random and (b) GCD workload trace.

adapting to the varying resource needs of VMs.

For the PlanetLab workload trace, VMCUP–M proposed in Publication IV significantly reduces the energy consumption while consolidating VMs for most of the VM selection policies under the THR (Figure 3.13a) and the LR (Figure 3.13b) hot thresholds. It incurs in an energy consumption that is 4.4% lower than MMT and 9.07% lower than MU; however, it has a slightly higher energy consumption than the MC and RS selection schemes. Thus happens as the PlanetLab VMs traces are mainly CPU–bound. Additionally, the memory utilization of VMs in PlanetLab is almost constant over time.

Figure 3.14 compares the performance of VMCUP–M against the BG algorithms in terms of energy consumption. VMCUP–M consumes less energy than both BG and BG–MUP over the THR and LR overutilized host detection approaches for both the random (Figure 3.14a) and the GCD (Figure 3.14b) workloads. These results are due to the fact that BG allocates the migrating VMs onto idle servers first because they have the lowest volume metric.

Number of Migrations and Power State Changes

Figure 3.15 depicts the number of migrations per VM. The THR (top part of the figures) and the LR (bottom part of the figures) hot threshold algorithms are separated into sub–figures. As it can be observed, VMCUP–M reduces the number of migrations by more than 65% (Figure 3.15a) for GCD; and by more than 92% for PlanetLab (Figure 3.15b) for both the THR and LR algorithms, respectively. The obtained results illustrate

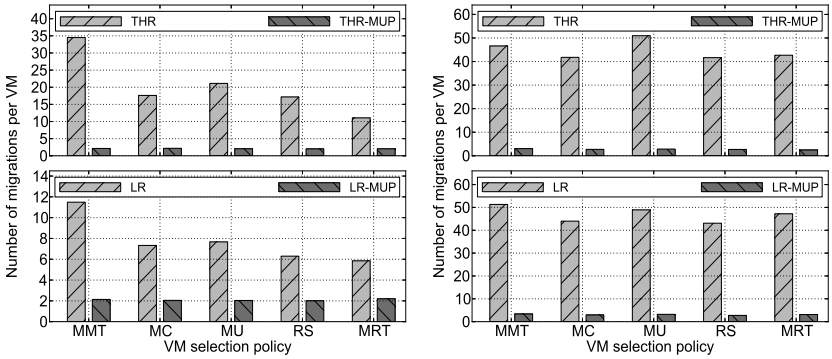


Figure 3.15. Number of migrations per VM under the THR and LR algorithms with different VM selection policies (Publication IV) for the: (a) GCD and (b) PlanetLab workload traces.

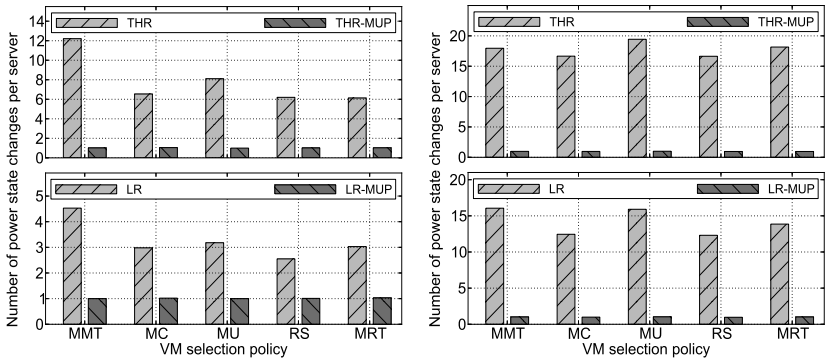


Figure 3.16. Number of power state changes per data center under the THR and LR algorithms with different VM selection policies (Publication IV) for the: (a) GCD and (b) PlanetLab workload traces.

that the algorithm with prediction reduces unnecessary migrations due to temporary resource load.

Figure 3.16 shows that the number of power state changes per data center with MUP is much smaller for all considered VM selection policies. The reduction in the server switches is more than 83% (THR) and 59% (LR) for GCD (Figure 3.16a); and by more than 91% for PlanetLab (Figure 3.16b). This is because VMCUP–M correctly forecasts underutilized servers, thus limiting the frequency of server switches from idle to a low–power state and vice versa. It is important to remember that hosts cannot perform any useful processing while changing their power states. These results are due to the fact that VMCUP–M correctly predicts overutilized

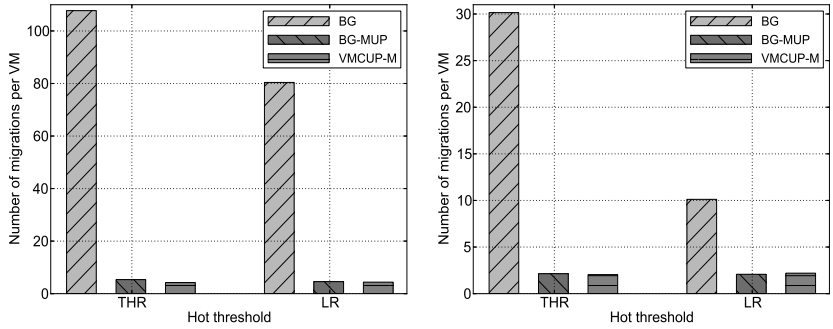


Figure 3.17. Number of migrations per VM of VMCUP–M and BG under the THR and LR algorithms (Publication IV) for the: (a) random and (b) GCD workload traces.

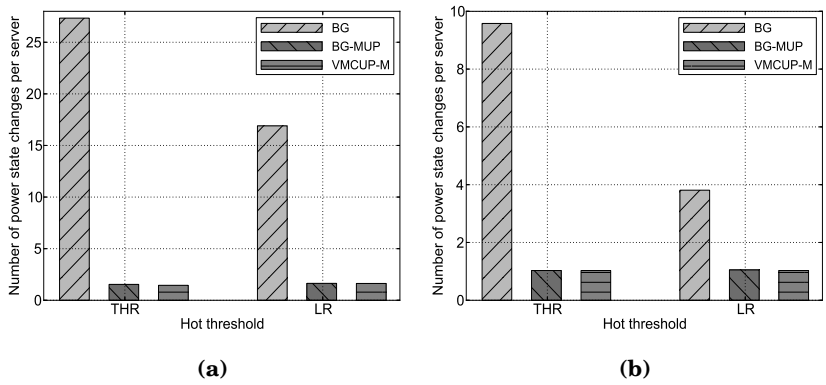
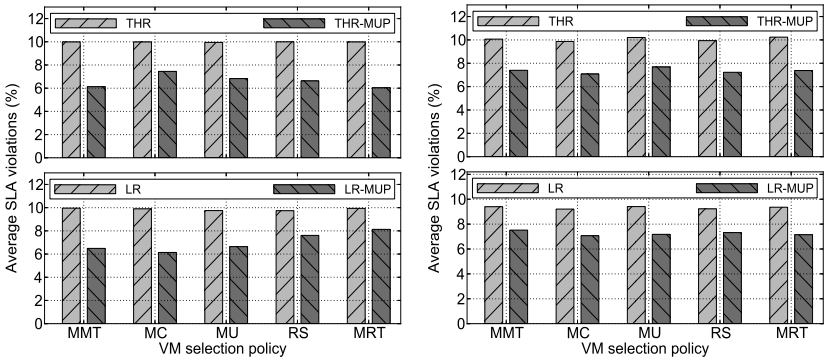


Figure 3.18. Number of power state changes per data center of VMCUP–M and BG under the THR and LR algorithms (Publication IV) for the: (a) random and (b) GCD workload traces.

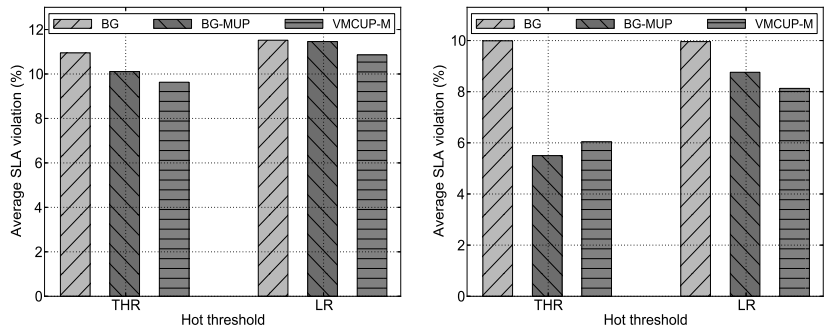
and underutilized servers based on current as well as future load. Furthermore, VMCUP–M also effectively finds the suitable destination hosts while evaluating VM migration, thus avoiding unnecessary migrations from the selected target hosts in near future.

Figure 3.17 and Figure 3.18 compare the performance of VMCUP–M against the BG algorithms for the random and GCD workload traces. In particular, VMCUP–M significantly reduces both the number of migrations and the number of power state changes per server. These results are due to the fact that VMCUP–M only selects the VMs that contribute more to the considered overloaded server according to its resource temperature. This helps avoid unnecessary migrations and power state changes.



(a) **(b)**

Figure 3.19. SLA compliance under the THR and LR algorithms with different VM selection policies (Publication IV) for the: (a) GCD and (b) PlanetLab workload traces.



(a) **(b)**

Figure 3.20. SLA compliance of VMCUP–M and BG under the THR and LR algorithms (Publication IV) for the: (a) random and (b) GCD workload traces.

SLA Compliance

The SLA compliance of VMCUP–M with the considered VM selection schemes is illustrated in Figure 3.19. The results show that VMCUP–M significantly reduces the average SLA violation percentage, i.e., by more than 16% (with the GCD workload, shown in Figure 3.19a); and more than 22% (with the PlanetLab workload, shown in Figure 3.19b) for both THR and LR. This ensures that the destination servers do not become overutilized while migrating VMs thanks to MUP. The reported results demonstrate that VMCUP–M performs well while consolidating VMs.

Finally, VMCUP–M also reduces the average SLA violation percentage (Figure 3.20), compared to BG algorithm by using MUP for the overloaded

and underloaded hosts. The reason is that the destination servers do not become overutilized in neither the current nor the future time period while migrating VMs.

4. An Application of Distributed Computing to Big Data Analytics

Modern web and distributed applications are facing an ever–growing demand for analyzing huge amounts of data. Wikipedia[1] is a meaningful example of a large Internet data source that is used for several applications, in particular, for those involving semantics. As of November 2015, the English version of Wikipedia included over 5 million articles with more than 37.8 million pages, and the total size of English articles exceeded 49 GB. The size of Wikipedia is rapidly growing by daily updates. As a consequence, it is time consuming to extract useful information from large amounts of data on a single workstation. Such processing requires novel approaches and technologies in order to cope with the complexity of big data analytics. This chapter presents research done in Publication V about a novel system for data–intensive applications, called Distributed Semantic Analysis (DSA). Specifically, DSA integrates a distributed–based approach with semantic analysis, thus enabling end users to efficiently process large amounts of data in a short time. Experimental results show two major improvements over the state of the art with particular reference to the Explicit Semantic Analysis (ESA) method [41]. First, DSA significantly reduces the computation time to process big data, thus enabling the use of larger inputs and minimizing the costs for using computing resources. Second, DSA obtains a higher correlation of semantic relatedness than existing solutions.

4.1 Distributed Semantic Analysis

The architecture of the DSA system for measuring semantic relatedness is illustrated in Figure 4.1. In particular, DSA uses the Mahout machine learning library [4] running on top of the Hadoop MapReduce distributed

[1] `http://www.wikipedia.org`

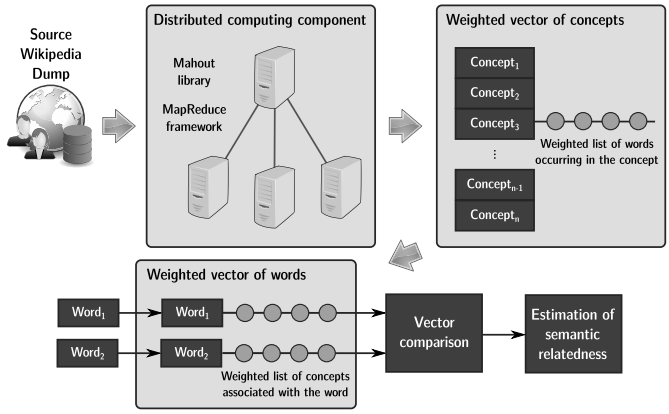


Figure 4.1. The Distributed Semantic Analysis (DSA) system for measuring semantic relatedness proposed in Publication V.

framework [39] to extract meaningful information from Wikipedia–based concepts and build weighted vectors. Wikipedia includes a wide range of articles about almost every subject by using human expertise in different areas. This motivates us to use Wikipedia data as the source semantic space in this dissertation. I would like to recall that the proposed DSA system — built on top of Hadoop MapReduce — is general enough to supports big data analytics with other forms of semantic knowledge such as WordNet, Flicker, and Twitter.

The following overviews the method used to build the weighted vectors from a full dump of Wikipedia–based concepts by using DSA.

4.1.1 Pre–Processing of Wikipedia Data

The first step is to take a full dump of Wikipedia articles in English[2] as input. The article texts are used as they provide the largest amount of knowledge. In the following, they will be just referred to as *articles* for brevity. DSA uses the WIKIPEDIATOSEQUENCEFILE class of Mahout to obtain the sequence file from the XML data format [109].

The strength of the association between a word and a concept can be expressed through the a single–term frequency inverse document frequency (TF–IDF) weight [109], namely, the value of term frequency (TF) multiplied by the inverse document frequency (IDF). Mahout also provides a class for creating TF–IDF term vectors from sequence files, along with several options to remove high–frequency features such as *stop words* (e.g., those with limited or ambiguous lexical meaning such as a, an, the,

[2] `https://dumps.wikimedia.org/enwiki/latest/`

who, what, are, is, and so on) and very low–frequency features (i.e., *rate words*).

After building the weighted vectors, all Wikipedia articles are represented as a set of associated $Concept : \{Term_1 : (tf - idf)Vector_1,\ Term_2 : (tf - idf)Vector_2,\ \ldots,\ Term_n : (tf - idf)Vector_n\}$. For instance, the concept vectors associated to an article entitled *Computer* in Wikipedia could be:

```
computer: {
    computer: 46.148487091,
    mouse: 12.498124122,
    keyboard: 11.278808593,
    memory: 18.02132034,
    ...
}
```

DSA shows one example of distributed processing jointly involving different types of heterogeneous resources. First, DSA must allow storing a large amount of input data. To support high throughput access to these data, DSA employs the Hadoop Distributed File System (HDFS) for storage purposes. Second, CPU and memory resources are used to build the weighted vectors of Wikipedia–based concepts and perform statistical analysis of large amounts of data in a short time. Indeed, improving the processing time of big data applications such as DSA leads to a significant cost reduction.

4.2 A Big Data Application

Semantic analysis is the process of of extracting high–level and language–independent meaning from syntactic structures [46]. It is a building block for several applications, especially those involving social network analytics. Semantic analysis is inherently data–intensive, thus, it may greatly benefit from the distributed computing paradigm. However, solution proposed in the literature have not fully exploited such an approach. For instance, one of the most important methods for semantic analysis based on Wikipedia is represented by ESA [41]. ESA uses text classification techniques that allow to explicitly represent the meaning of any text in terms of Wikipedia–based concepts. However, ESA employs machine learning

techniques to build concept vectors on a standard workstation, and does not scale to multiple hosts. As a consequence, the source Wikipedia data (in form of an XML dump) requires a pre–processing step to remove non–relevant information so as to obtain a smaller input (about 2.9 GB) that can be then processed on a single machine.

Publication V presents a large–scale data processing for semantic analysis that leverages the proposed DSA system. In particular, a fully automated process is proposed to measure semantic relatedness between two words. In such a scenario, the proposed method is based on concept vectors extracted from full dump of Wikipedia articles as knowledge–based information thanks to distributed computing. The following sections evaluate and summarize the performance study and comparison with the state of the art conducted in Publication V.

4.2.1 Word Semantic Relatedness

The semantic relatedness between words is not measured directly, but it is rather determined through a set of concepts highly related to them. DSA calculates the semantic relatedness between words w_1 and w_2 through the two steps below.

- Determining the concepts (or articles) of Wikipedia, which are related to words w_1 and w_2. In detail, w_1 is mapped to *concept:(tf–idf)vector* $L(w_1) = \{(C_1^1,\ V_1^1),\ (C_2^1,\ V_2^1),\ (C_3^1,\ V_3^1),\ ...,\ (C_M^1,\ V_M^1)\}$ and w_2 is mapped to *concept:(tf–idf)vector* $L(w_2) = \{(C_1^2,\ V_1^2),\ (C_2^2,\ V_2^2),\ (C_3^2,\ V_3^2),\ ...,\ (C_N^2,\ V_N^2)\}$ ($M < N$).

- Calculating the semantic relatedness between two words w_1 and w_2 by using the *Word Semantic Relatedness* (namely, WSRel) metric

$$\mathrm{WSRel}(w_1, w_2) = \mathrm{WSRel}(L(w_1), L(w_2)) = \frac{\sum_{i=1}^{N} V_i^1 \cdot V_i^2}{\sqrt{\sum_{j=1}^{M} \left(V_j^1\right)^2} \cdot \sqrt{\sum_{l=1}^{N} \left(V_l^2\right)^2}}.$$

  The $\mathrm{WSRel}(w_1, w_2)$ values range from 0 (i.e., no semantic relatedness) to 1 (i.e., perfect semantic relatedness).

4.2.2 Summary of Results

In this dissertation, DSA was realized on top of a Hadoop MapReduce cluster of five physical machines. One of them was used as the master

Method	M&C	R&G	WS–353
WikiRelate! [131] (Wikipedia February 2006)	0.49	0.56	Full: 0.48 Test: 0.55
Wikipedia knowledge–based [112] (Wikipedia May 2007)	0.57	0.61	Full: 0.54 Test: 0.64
Wikipedia link–based [95] (Wikipedia November 2007)	0.70	0.6	0.69
Wikipedia snippet–based [132]	0.8	0.797	n/a
ESA [41, 42] (Wikipedia March 2006)	0.73	0.82	0.75
TSA [117]	n/a	n/a	0.8
LSAC [72]	0.764	0.715	Full: 0.612 Test: 0.759
Implementation of LSA in [38]	n/a	n/a	0.56
WikiRelate! [131] (WordNet 2.1)	0.82	0.86	Full: 0.36 Test: 0.39
DSA (Wikipedia February 2012)	**0.810**	**0.806**	Full: **0.835** Test: **0.849**

Table 4.1. Pearson's correlation coefficient of the different approaches for the considered benchmark datasets in Publication V.

node while the other four served as slave nodes. Each machine had 2 cores (3.2 GHz per core), 8 GB memory and 1.2 TB storage capacity. The used software was Ubuntu Linux 10.04, Hadoop (version 0.20.2) and the Mahout machine learning library (version 0.6). The nodes of the cluster were connected through a local area network.

I would like to emphasize that cloud data centers offer heterogeneous computing resources in the variety of different types of virtual machine (VM) instances, which allows end users to set up and maintain a large–scale Hapdoop MapReduce cluster. To support data–intensive applications, several cloud providers have actually included the MapReduce framework within VMs in their data centers, for instance, as Google MapReduce and Amazon Elastic MapReduce. This allows end users to deploy a variety of Hadoop clusters based on different VM instance types.

Due to the scalable approach proposed, the total time to build weighted vectors from the XML dump of Wikipedia articles in English took about 43 minutes. As a reference, the weighted vectors with the same input were also built on a single system, and the total processing time took about 5.5 hours, which is consistent with the results in [42]. Therefore, using the DSA system is an efficient means to reduce the computation time and improve the system performance.

Processing a large data set indeed enables a more accurate semantic analysis. Table 4.1 shows the semantic relatedness, expressed through

the Person's correlation coefficient (the higher the better), under different datasets for several methods available in the literature and DSA. In the table, the methods are grouped according to their characteristics. The first group is represented by the methods exploiting Wikipedia, namely: WikiRelate! [131]; knowledge–based measure [112]; link–based measure [95]; snippet–based measure [132]; ESA [41, 42]; and its extension Temporal Semantic Analysis (TSA) [117]. The second group is represented by methods relying on latent semantic analysis [73] (i.e., the experiments using LSAC [72] and the LSA implementation in [38]) and/or exploiting WordNet as input (i.e., WikiRelate! [131]). Finally, DSA is shown as a separate group in the table.

The results reported in Table 4.1 show that, over the M&C dataset, DSA performs better than all other approaches, except for WikiRelate! when using WordNet 2.1 as input. Specifically, DSA obtains a higher correlation coefficient than the solutions using Wikipedia as input. For the R&G dataset, DSA performs better than most of the other solutions. Specifically, the correlation coefficient of DSA is 0.806, that is only slightly lower than that of ESA (i.e., 0.82) and not so distant from that of WikiRelate (i.e., 0.86). As for the WS–353 dataset, DSA obtained the highest correlation coefficient among all considered approaches, even when using the test dataset. Furthermore, switching from the test to the full dataset yields a non–marginal increase of the correlation efficient, that achieves the remarkable value of 0.849. This value is very high especially if compared to that of the solutions that performed better than DSA in the R&G approach, namely, ESA (i.e., 0.75) and WikiRelate! (i.e., 0.39). In detail, the improvement of DSA over ESA is of 10.9% for the M&C dataset, and of 11.3% for the WS–353 dataset.

DSA with Wikipedia as semantic space achieved an improvement of 131.9% (full dataset) and 117.7% (test dataset) over WS–353 with respect to WordNet as semantic–based measure. The correlation coefficient of the WordNet–based measure is very low for WS–353 because in that dataset there are some word pairs containing at least one word that is not present in WordNet [112]. Another reason behind DSA performing better than WordNet for the WS–353 dataset is because the proposed approach models semantic relatedness rather than semantic similarity, while WordNet is designed to quantify the latter.

5. Conclusion

This chapter presents the contributions of this dissertation and discusses future research directions.

5.1 Contributions

IaaS clouds offer heterogeneous computing resources in the form of VM instances, which allows end users to lease resources and only pay for those that are actually used. To support the growing number of requested VMs, cloud providers are now building an increasing number of large–scale data centers. Managing such data centers and large amounts of VMs is indeed a challenging task. In fact, it involves VM management algorithms that are not only required to balance loads across multiple resources but also to operate at lower energy consumption levels. Both objectives are achieved by optimizing the assignment of VMs to physical machines through efficient VM placement and consolidation algorithms, which are capable of: (1) limiting number of active physical servers; (2) creating idle servers, then transitioning such idle servers in power–saving states and reactivating them once required; and (3) minimizing the number of migrations and server switches. Furthermore, the performance of big data applications deployed in virtualized environments should be considered as the highest–layer of the software stack while managing VMs. Accordingly, a system that leverages distributed computing is designed to quickly extract meaningful information from large amounts of data and support big data analytics. This dissertation investigated the challenge of designing, implementing, and evaluating efficient VM management in IaaS cloud data centers containing heterogeneous physical servers and virtualized resources, with focus on energy efficiency. Specifically, the following are the three major contributions of this dissertation.

Virtual Machine Placement for Load Balancing across Multiple Resources in Cloud Data Centers (Publication I). VM placement aims at optimizing the assignment of VMs to physical machines, starting from the VM requests, with the goal of minimizing the number of active servers. Existing VM placement algorithms are mostly constrained to a limited number of resource types (e.g., CPU), thus resulting in unbalanced load or in the unnecessary activation of physical servers. To solve these limitations, a VM placement algorithm called Max–BRU was proposed to improve the resource utilization and to spread the load across diverse resources, including CPU, memory, storage, and network bandwidth. The Max–BRU algorithm not only optimizes resource utilization but also balances the usage of resources across multiple dimensions, thus reducing the number of active physical servers in a IaaS cloud data center. Simulation results showed two major improvements over the state of the art for VM placement. First, Max–BRU increases the resource utilization by minimizing the amount of physical servers used. Second, Max–BRU effectively balances the utilization of multiple types of resources.

Virtual Machine Consolidation for Energy–Efficient Cloud Data Centers (Publication II, Publication III, and Publication IV). VM consolidation aims at reducing the number of active servers in a data center so as to reduce the total power consumption. In this context, most of the existing solutions rely on aggressive VM migration, thus resulting in unnecessary overhead and energy wastage. Moreover, VM consolidation should take into account multiple resource types simultaneously, since CPU is not the only critical resource in cloud data centers. In fact, also memory and network bandwidth may become a bottleneck, possibly causing violations in the SLA. To tackle both these issues, this dissertation has proposed two VM consolidation algorithms, called MRS (Publication II) and VMCUP–M (Publication IV), for improving the energy efficiency of cloud data centers. In this context, the proposed VM consolidation algorithms: (1) remove resource fragmentation of servers after a number of VMs have been added to (or removed from) the cloud data center as well as when VM workloads increase or decrease over time; (2) create idle servers and switch them into a low–power state to save energy; and (3) minimize the number of VM migrations and server switches along with the number of active servers. The goals above were achieved by integrating VM consolidation algorithms with an efficient overloaded and underloaded host management scheme. Such an integration allowed for a re-

liable characterization of hot and cold servers, thus limiting the number of unnecessary VM migrations and server switches along with the number of active machines. Simulation results on both synthetic and real–world workloads showed that, in comparison with the state of the art, consolidation with efficient hot and cold spot management enables reducing the number of migrations and power state changes while complying with the SLA. Furthermore, thanks to the proposed multiple usage prediction (MUP) scheme, our solution is able to cooperate with existing VM selection policies to lower the energy consumption, limit the frequency of VM migrations and server switches in a data center. MUP also inter–operates with existing VM placement algorithm (i.e., power–aware best fit decreasing) to correctly select the target host that does not become a hot spot in the long–term future.

A Distributed Computing Solution for Big Data Analytics (Publication V). Processing huge amounts of data consumes significant energy because it is time–consuming to extract meaningful information from them. This dissertation has also proposed an efficient approach, called Distributed Semantic Analysis (DSA), that integrated distributed computing with semantic analysis so as to process large amounts of data in a scalable manner. In particular, DSA targeted application–specific requirements (e.g., response time) while performing the heavy–duty tasks. A testbed evaluation showed that DSA significantly reduces the computation time to analyze big data, thus also decreasing the associated energy consumption. In particular, DSA is able to process large amounts of semantic data, for instance, a full dump of Wikipedia articles, without resorting to preliminary filtering of the source data.

5.2 Future Research Directions

The work in this dissertation can be further developed along multiple directions.

The first improvement targets VM consolidation. Publication II, Publication III, and Publication IV have extensively discussed VM consolidation algorithms that execute periodically according to a predefined consolidation interval (i.e., every five minutes) to eliminate overload situations or create idle physical servers. However, setting static consolidation intervals is not an effective means for IaaS cloud environments with dynamic workloads, in which the utilization of VMs running on a physical server

continuously changes over multiple resource dimensions. This may yield a significant performance degradation when VM consolidation is executed during periods of high utilization of any resource dimension [124]. It is therefore important to investigate approaches to accurately estimate the consolidation times based on the history of VM workloads.

The second improvement concerns network–related aspects, especially, the data center network and that of interacting VMs [92]. Publication I and Publication II have taken into account network bandwidth as one dimension in the proposed VM placement and consolidation algorithms. However, they did not consider the network topologies of the data center and of the deployed VMs. Therefore, VM management algorithms could be extended to take the data center network topologies into account while computing the assignment between the virtual and the physical machines [64]. This would allow to reduce the VM live migration time and the energy consumption by selecting the network links with the best performance and energy trade–offs. Furthermore, cloud users may request several VMs with massive inter–VM communication (e.g., running web servers and databases). Therefore, VM management algorithms could further reduce the network communication costs by explicitly considering the communication between VMs, i.e., by placing the set of communicating VMs on the same or on closely–located physical machines [18].

The third improvement targets VM management approaches over multiple IaaS clouds. Cloud providers recently started to deploy large–scale data centers consisting of multiple server farms, possibly distributed in geographically different locations [20, 48, 80]. Publication III and Publication IV have extended the CloudSim simulation toolkit to support multiple resource types and simulated a single cloud data center. However, the VM management algorithms proposed in this dissertation should be further extended for data centers spanning across multiple locations.

The fourth improvement involves evaluating the performance of the proposed VM management algorithms in real cloud data centers and comparing the related performance with existing solutions in open–source IaaS cloud management systems (e.g., OpenStack). As the target system is an IaaS, conducting large–scale experiments on a real infrastructure is extremely difficult [10]. However, the distributed computing system for big data analytics presented in Publication V was deployed and evaluated in a real testbed environment. A more extensive analysis could be conducted on a large–scale scenario as a future work.

Bibliography

[1] Mansoor Alicherry and T.V. LakShman. Network aware resource allocation in distributed clouds. In *Proceedings of the 31st Conference on Computer Communications (INFOCOM '12)*, pages 963–971, 2012.

[2] Jorn Altmann, Matthias Hovestadt, and Odej Kao. Business support service platform for providers in open cloud computing markets. In *Proceedings of the 7th International Conference on Networked Computing (INC)*, pages 149–154, 2011.

[3] Amazon. Amazon Elastic Compute Cloud (EC2). `http://aws.amazon.com/ec2/`, 2015.

[4] Apache Software Foundation. Mahout Machine Learning Library. `http://mahout.apache.org/`, 2012.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Communications of the ACM*, 53:50–58, 2009.

[6] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM symposium on Operating systems principles (SOSP '03)*, pages 164–177, 2003.

[7] Luiz André Barroso and Urs Hölzle. The case for energy–proportional computing. *Computer*, 40:33–37, 2007.

[8] Luiz André Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse–Scale Machines*, page 108 pages. Morgan and Claypool Publishers, 2009.

[9] Sobir Bazarbayev, Matti Hiltunen, Kaustubh Joshi, William H. Sanders, and Richard Schlichting. Content–based scheduling of virtual machines in the cloud. In *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 93–101, 2013.

[10] Anton Beloglazov. *Energy–Efficient Management of Virtual Machines in Data Centers for Cloud Computing*, page 239 pages. PhD Thesis. The University of Melbourne, 2013.

[11] Anton Beloglazov and Rajkumar Buyya. Managing overload hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions of Parallel and Distributed Systems*, 24:1366–1379, 2012.

[12] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24:1397–1420, 2012.

[13] Shshil Bhardwaj, Leena Jain, and Sandeep Jain. Cloud computing: A study of infrastructure as a service (iaas). *International Journal of Engineering and Information Technology*, 2:60–63, 2010.

[14] Ricardo Bianchini and Ram Rajamony. Power and energy management for server systems. *IEEE Computer*, 37:68–74, 2004.

[15] Robert Birke, Lydia Y. Chen, and Evgenia Smirni. Data centers in the wild: A large performance study. Technical report, IBM Research, April 2012.

[16] Mark Blackburn. Five ways to reduce data center server power consumption. Technical report, The Green Grid, 2008.

[17] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM '07)*, pages 119–128, 2007.

[18] Sumit Kumar Bose, Scott Brock, Ronald Skeoch, and Shrisha Rao. Cloudspider: Combining replication with scheduling for optimizing live migration of virtual machines across wide area networks. In *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 13–22, 2011.

[19] Tracy D. Braun, Howard Jay Siegel, Noah Beck, Ladislau L. Bölöni, Muthucumaru Maheswaran, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, and Bin Yao. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 6:810–837, 2001.

[20] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. Intercloud: Utility–oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP '10)*, pages 13–31, 2010.

[21] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Journal of Future Generation Computer Systems*, 25:599–616, 2009.

[22] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, De Rose, César A. F. De Rose, and Rajkumar Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource

provisioning algorithms. *Software Practice and Experiment*, 41(1):23–50, 2011.

[23] Michael Cardosa, Madhukar R. Korupolu, and Aameek Singh. Shares and utilities based power consolidation in virtualized server environments. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management (IM '09)*, pages 327–334, 2009.

[24] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5:164–177, 2012.

[25] Kyle Chard, Simon Caton, Omer Rana, and Kris Bubendorfer. Social cloud: Cloud computing in social networks. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD)*, pages 99–106, 2010.

[26] Wei Chen, Xiaoqiang Qiao, Jun Wei, and Tao Huang. A profit–aware virtual machine deployment optimization framework for cloud platform providers. In *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD)*, pages 17–24, 2012.

[27] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation (NSDI'05)*, pages 273–286, 2005.

[28] Thiago Cordeiro, Douglas Damalio, Nadilma Pereira, Patricia Endo, André Palhares, Glauco Gonçalves, Djamel Sadok, Judith Kelner, Bob Melander, Victor Souza, and Jan-Erik Mangs. Open source cloud computing platforms. In *Proceedings of the 9th IEEE International Conference on Grid and Cooperative Computing (GCC)*, pages 366–371, 2010.

[29] Antonio Corradi, Mario Fanelli, and Luca Foschini. Vm consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 32:118–127, 2014.

[30] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Journal of Communications of the ACM*, 51:107–113, 2008.

[31] Umesh Deshpande, Xiaoshuang Wang, and Kartik Gopalan. Live gang migration of virtual machines. In *Proceedings of the 20th international symposium on High performance distributed computing (HPDC '11)*, pages 135–146, 2011.

[32] Jaliya Ekanayake and Geoffrey Fox. High performance parallel computing with clouds and cloud technologies. *Cloud Computing. The series Lecture Notes of the Institute for Computer Sciences, Social–Informatics and Telecommunications Engineering*, 34:20–38, 2010.

[33] David Erickson, Brandon Heller, Shuang Yang, Jonathan Chu, Jonathan Ellithorpe, Scott Whyte, Stephen Stuart, Nick McKeown, Guru Parulkar, and Mendel Rosenblum. Optimizing a virtualized data center. In *Proceedings of the ACM SIGCOMM 2011 conference (SIGCOMM '11)*, pages 478–479, 2011.

[34] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse–sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA '07)*, pages 13–23, 2007.

[35] Eugen Feller. *Autonomic and Energy–Efficient Management of Large–Scale Virtualized Data Centers*, page 180 pages. Distributed, Parallel, and Cluster Computing [cs.DC]. PhD Thesis. Université Rennes 1, 2012.

[36] Eugen Feller, Christine Morin, and Armel Esnault. A case for fully decentralized dynamic vm consolidation in clouds. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 26–33, 2012.

[37] Tiago C. Ferreto, Marco A. S. Netto, Rodrigo N. Calheiros, and César A. F. De Rose. Server consolidation with migration control for virtualized data centers. *Journal of Future Generation Computer System*, 27:1027–1034, 2011.

[38] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131, 2002.

[39] Apache Software Foundation. Apache Hadoop. `http://hadoop.apache.org/`, 2015.

[40] The Apache Software Foundation. CloudStack: Open Source Cloud Computing. `http://cloudstack.apache.org/`, 2015.

[41] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia–based explicit semantic analysis. In *Proceedings of the 21th National Conference on Artificial Intelligence*, pages 1606–1611, 2007.

[42] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia–based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research*, 34:443–498, 2009.

[43] Saurabh Kumar Garg, Rajkumar Buyya, and Howard Jay Siegel. Time and cost trade–off management for scheduling parallel applications on utility grids. *Journal of Future Generation Computer Systems*, 91:151–160, 2009.

[44] Peter Garraghan, Paul Townend, and Jie Xu. An analysis of the server characteristics and resource utilization in google cloud. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E '13)*, pages 124–131, 2013.

[45] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Resource pool management: Reactive versus proactive or let's be friends. *The International Journal of Computer and Telecommunications Networking*, 53:2905–2922, 2009.

[46] C. Goddard. *Semantic analysis: A practical introduction*. Oxford University Press, second edition, 2011.

[47] Google. Compute Engine: Run large–scale workloads on virtual machines hosted on Google's infrastructure. Choose a VM that fits your needs and gain the performance of Google's worldwide fiber network. `https://cloud.google.com/products/compute-engine`, 2015.

[48] Hadi Goudarzi. *SLA–based, Energy–Efficient Resource Management in Cloud Computing Systems*, page 173 pages. PhD Thesis. University of Southern California, 2013.

[49] Hadi Goudarzi, Mohammad Ghasemazar, and Massoud Pedram. Sla–based optimization of power and migration cost in cloud computing. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 172–179, 2012.

[50] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 39:63–73, 2009.

[51] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy–aware server provisioning. In *Proceedings of the 30st Conference on Computer Communications (INFOCOM '11)*, pages 1332–1340, 2011.

[52] Abhishek Gupta, Laxmikant V. Kalé, Dejan Milojicic, Paolo Faraboschi, and Susanne M. Balle. Hpc–aware vm placement in infrastructure clouds. In *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, pages 11–20, 2013.

[53] J. Octavio Gutierrez-Garcia and Kwang Mong Sim. A family of heuristics for agent–based elastic cloud bag–of–tasks concurrent scheduling. *Journal of Future Generation Computer Systems*, 29:1682–1699, 2012.

[54] James Hamilton. Cooperative expendable micro–slice servers (cems): Low cost, low power servers for internet–scale services. In *Proceedings of the 4th Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 1–8, 2009.

[55] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)*, pages 41–50, 2009.

[56] Michael R. Hines and Kartik Gopalan. Post–copy based live virtual machine migration using adaptive pre–paging and dynamic self–ballooning. In *Proceedings of the ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE '09)*, pages 51–60, 2009.

[57] HP. HP Helion Eucalyptus: Open source hybrid cloud software for AWS users. `http://www8.hp.com/us/en/cloud/helion-eucalyptus-overview.html`, 2015.

[58] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, pages 89–96, 2010.

[59] Chris Hyser, Bret McKee, Rob Gardner, and Brian J. Watson. Autonomic virtual machine placement in the data center. Technical report, Hewlett Packard Laboratories, February 2008.

[60] Chris Hyser, Bret McKee, Rob Gardner, and Brian J. Watson. Autonomic virtual machine placement in the data center. Technical report, Hewlett Packard Laboratories, February 2008.

[61] IBM. IBM Cloud. `http://www.ibm.com/us/en/`, 2015.

[62] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Journal of Future Generation Computer Systems*, 28:155–162, 2012.

[63] James Pearn. How many servers does Google have? `https://plus.google.com/+JamesPearn/posts`, 2012.

[64] Joe Wenjie Jiang, Tian Lan, Sangtae Ha, Minghue Chen, and Mung Chiang. Join vm placement and routing for data center traffic engineering. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications: Mini–Conference*, pages 2876–2880, 2012.

[65] Hao Jin, Deng Pan, Jing Xu, and Niki Pissinou. Efficient vm placement with multiple deterninistic and stochastic resources in data centers. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pages 2505–2510, 2012.

[66] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Ottawa Linux Symposium*, pages 225–230, 2007.

[67] Thomas Knauth. *Energy Efficient Cloud Computing: Techniques and Tools*, page 147 pages. PhD Thesis. Technische Universität Dresden, 2014.

[68] Thomas Knauth and Christof Fetzer. Energy–aware scheduling for infrastructure clouds. In *Proceedings of the 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 58–65, 2012.

[69] Jonathan G. Koomey. Estimating total power consumption by servers in the u.s. and the world. Technical report, Lawrence Berkeley National Laboratory, 2007.

[70] Jonathan G. Koomey. Growth in data center electricity use 2005 to 2010. Technical report, Analytics Press, August 2011.

[71] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12:1–15, 2009.

[72] Darrell Laham. Latent Semantic Analysis@CU Boulder. `http://lsa.colorado.edu/`, visited January 30, 2012.

[73] T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis. *Discourse Process*, 25(2-3):259–284, 1998.

[74] Gunho Lee. *Resource Allocation and Scheduling in Heterogeneous Cloud Environments*, page 113 pages. PhD Thesis. University of California, Berkeley, 2012.

[75] Sangmin Lee, Rina Panigrahy, Vijayan Prabhakaran, Venugopalan Ramasubramanian, Kunal Talwar, Lincoln Uyeda, and Udi Weider. Validating heuristics for virtual machines consolidation. Technical Report MSR-TR-2011-9, Microsoft Research, January 2011.

[76] Young Choon Lee and Albert Y. Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60:268–280, 2012.

[77] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Server–level power control. In *Proceedings of the 4th International Conference on Autonomic Computing (ICAC '07)*, page 4, 2007.

[78] Bo Li, Jianxin Li, Jinpeng Huai, Tianyu Wo, Qin Li, and Liang Zhong. Enacloud: An energy–saving application live placement approach for cloud computing environments. In *Proceedings of the IEEE International Conference on Cloud Computing (CLOUD)*, pages 17–24, 2009.

[79] Wubin Li. *Algorithms and Systems for Virtual Machine Scheduling in Cloud Infrastructures*, page 136 pages. PhD Thesis. Umeå University, 2014.

[80] Wubin Li, Johan Tordsson, and Erik Elmroth. Modeling for dynamic cloud scheduling via migration of virtual machines. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 163–171, 2011.

[81] Wubin Li, Johan Tordsson, and Erik Elmroth. Virtual machine placement for predictable and time–constrained peak loads. *Economics of Grids, Clouds, Systems, and Services*, 7150 of the series Lecture Notes in Computer Science:120–134, 2012.

[82] Xin Li, Zhuzhong Qian, Ruiqing Chi, Bolei Zhang, and Sanglu Lu. Balancing resource utilization for continuous virtual machine requests in clouds. In *Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 266–273, 2012.

[83] Xin Li, Zhuzhong Qian, Sanglu Lu, and Jie Wu. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in data center. *Mathematical and Computer Modelling*, 58:1222–1235, 2013.

[84] Ching Chi Lin, Pangfeng Liu, and Jan Jan Wu. Energy–efficient virtual machine provison algorithms for cloud systems. In *Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing (UCC)*, pages 81–88, 2011.

[85] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right–sizing for power–proportional data centers. In *Proceedings of the 30st Conference on Computer Communications (INFOCOM '11)*, pages 1098–1106, 2011.

[86] Xuan Lin, Anwar Mamat, Ying Lu, Jitender Deogun, and Steve Goddard. Real–time scheduling of divisible loads in cluster computing environments. *Journal of Parallel and Distributed Computing*, 70:296–308, 2009.

[87] Liang Liu, Hao Wang, Xue Liu, Xing Jin, WenBo He, QingBo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session (ICAC-INDST'09)*, pages 29–38, 2009.

[88] Ali José Mashtizadeh, Min Cai, Gabriel Tarasuk-Levin, Ricardo Koller, Tal Garfinkel, and Sreekanth Setty. Xvmotion: Unified virtual machine migration over long distance. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC 14)*, pages 97–108, 2014.

[89] Carlo Mastroianni, Michela Meo, and Giuseppe Papuzzo. Probabilistic consolidation of virtual machines in self–organizing cloud data centers. *IEEE Transactions on Cloud Computing*, 1:125–228, 2013.

[90] Vimal Mathew, Ramesh K. Sitaraman, and Prashant Shenoy. Energy–aware load balancing in content delivery networks. In *Proceedings of the 31st Conference on Computer Communications (INFOCOM '12)*, pages 954–962, 2012.

[91] David Meisner, Brian T. Gold, and Thomas F. Wenisch. Powernap: eliminating server idle power. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '09)*, pages 205–216, 2009.

[92] Xiaoqiao Meng, Vasileios Pappas, and Li Zhang. Improving the scalability of data center networks with traffic–aware virtual machine placement. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM '10)*, pages 1154–1162, 2010.

[93] Microsoft. Microsoft Hyper–V. `https://technet.microsoft.com/en-us/windowsserver/dd448604.aspx`, 2015.

[94] K. Mills, J. Filliben, and C. Dabrowski. Comparing vm–placement algorithms for on–demand clouds. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 91–98, 2011.

[95] David Milne and Ian H. Witten. An effective, low–cost measure of semantic relatedness obtained from Wikipedia links. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI '08)*, pages 25–30, 2008.

[96] Yue Minyi. A simple proof of the inequality ffd (l) < 11/9 opt (l) + 1, for all l for the ffd bin–packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7:321–331, 1991.

[97] Mayank Mishra and Anirudha Sahoo. On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In *Proceedings of the 4th IEEE International Conference on Cloud Computing (CLOUD*, pages 275–282, 2011.

[98] Aziz Murtazaev and Sangyoon Oh. Sercon: Server consolidation algorithm using live migration of virtual machines for green computing. *IETE Technical Review*, 28:212–231, 2011.

[99] Susanta Nanda and Tzi-cker Chiueh. A survey on virtualization technologies. Technical report, Experimental Computer Systems Lab, State University of New York, February 2005.

[100] Trung Hieu Nguyen, Mario Di Francesco, and Antti Ylä-Jääski. Extracting knowledge from wikipedia articles through distributed semantic analysis. In *Proceedings of the 13th International Conference on Knowledge Management and Knowledge Computing (i-KNOW '13)*, pages 188–195, 2013.

[101] Trung Hieu Nguyen, Mario Di Francesco, and Antti Ylä-Jääski. A multi–resource selection scheme for virtual machine consolidation in cloud data centers. In *Proceedings of the 6th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 234–239, 2014.

[102] Trung Hieu Nguyen, Mario Di Francesco, and Antti Ylä-Jääski. A virtual machine placement algorithm for balanced resource utilization in cloud data centers. In *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD)*, pages 474–481, 2014.

[103] Trung Hieu Nguyen, Mario Di Francesco, and Antti Ylä-Jääski. Virtual machine consolidation with usage prediction for energy–efficient in cloud data centers. In *Proceedings of the 8th IEEE International Conference on Cloud Computing (CLOUD)*, pages 750–757, 2015.

[104] Dusit Niyato, Sivadon Chaisiri, and Lee Bu Sung. Optimal power management for server farm to support green computing. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '09)*, pages 84–91, 2009.

[105] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graiano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. The eucalyptus open–source cloud computing system. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, pages 17–24, 2009.

[106] University of Chicago. Nimbus is cloud computing for science. `http://nimbusproject.org`, 2015.

[107] OpenNebula. `http://www.opennebula.org`, 2015.

[108] Oracle. VirtualBox. `https://www.virtualbox.org/`, 2015.

[109] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. Manning Publications, 2012.

[110] Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, and Kang G. Shin. Performance evaluation of virtualization technologies for server consolidation. Technical report, HP Laboratories Technical Report, April 2007.

[111] KyoungSoo Park and Vivek S. Pai. Comon: A mostly–scalable monitoring system for planetlab. *SIGOPS Oper. Syst. Rev.*, 40(1):65–74, 2006.

[112] Simone Paolo Ponzetto and Michael Strube. Knowledge derived from Wikipedia for computing semantic relatedness. *Journal of Artificial Intelligence Research*, 30:181–212, 2007.

[113] The OpenStack Project. OpenStack: The Open Source Cloud Operating System. http://www.openstack.org/software/, 2015.

[114] QinZheng and Bharadwaj Veeravalli. Utilization based pricing for power management and profit optimization in data centers. *Journal of Parallel and Distributed Computing (JPDC)*, 72:27–34, 2012.

[115] Rackspace. Hosting Reports Third Quarter. http://ir.rackspace.com/phoenix.zhtml?c=221673&p=irol-newsArticle&ID=1988010, 2014.

[116] Rackspace. http://www.rackspace.com/, 2015.

[117] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on Word Web Wide*, pages 337–346, 2011.

[118] Dianne Rice, Jason Glick, Diana Cercy, Cathy Sandifer, and Bob Cramblitt. Standard Performance Evaluation Corporation (SPEC). https://www.spec.org/power_ssj2008/results/, 2015.

[119] RIKEN Integrated Cluster of Clusters (RICC). http://accc.riken.jp/ricc_e/, 2015.

[120] Pierre Riteau, Christine Morin, and Thierry Priol. Shrinker. Improving live migration of virtual clusters over wans with distributed data deduplication and content–based addressing. *EuroPar 2011 Parallel Processing, Lecture Notes in Computer Science*, 6852:431–442, 2011.

[121] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transaction on Cloud Computing*, 2:222–235, 2010.

[122] Mendel Rosenblum and Tal Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38:39–47, 2005.

[123] Sebastian Anthony. Google Compute Engine: For $2 million/day, your company can run the third fastest supercomputer in the world. http://www.extremetech.com/extreme/131962-google-compute-engine-for-2-millionday-your-company-can-\run-the-third-fastest-supercomputer-in-the-world, 2012.

[124] Thomas Setzer and Alexander Stage. Decision support for virtual machine reassignments in enterprise data centers. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps)*, pages 88–94, 2010.

[125] Aameek Singh, Madhukar R. Korupolu, and Dushmanta Mohapatra. Server–storage virtualization: Integration and load balancing in data centers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2008.

[126] Jame E. Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38:32–38, 2005.

[127] Biao Song, M.M. Hassan, and Eui nam Huh. A novel heuristic–based task selection and allocation framework in dynamic collaborative cloud service platform. In *Proceedings of the 2nd International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 360–367, 2010.

[128] Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 99:2647–2660, 2013.

[129] Benjamin Speitkamp and Martin Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on Services Computing*, 3:266–278, 2010.

[130] Alexander Stage and Thomas Setzer. Network–aware migration control and scheduling of differentiated virtual machine workloads. In *Proceedings of the IEEE Workshop on Software Engineering Challenges of Cloud Computing*, pages 9–14, 2009.

[131] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using Wikipedia. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1419–1424, 2006.

[132] Sheetal A. Takale and Sushma S. Nandgaonkar. Measuring semantic similarity between words using web documents. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 1(4):78–85, 2010.

[133] Ibrahim Takouna, Esra Alzaghoul, and Christoph Meinel. Robust virtual machine consolidation for efficient energy and performance in virtualized data centers. In *Proceedings of the IEEE International Conference on Green Computing and Communications (GreenCom)*, pages 470–477, September 2014.

[134] Selome Kostentinos Tesfatsion, Eddie Wadbro, and Johan Tordsson. A combined frequency scaling and application elasticity approach for energy–efficient cloud computing. *Sustainable Computing: Informatics and Systems. Special Issue on Energy Aware Resource Management and Scheduling*, 4:205–214, 2014.

[135] Wenhong Tian, Yong Zhao, Yuanliang Zhong, Minxian Xu, and Chen Jing. A dynamic and integrated loadbalancing scheduling algorithm for cloud datacenters. In *Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, pages 311–315, 2011.

[136] Timothy Prickett Morgan. A Rare Peek Into The Massive Scale of AWS. `http://www.enterprisetech.com/2014/11/14/rare-peek-massive-scale-aws/`, 2014.

[137] Traces of Google Workloads. `http://code.google.com/p/googleclusterdata/`, 2015.

[138] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39:50–55, 2008.

[139] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264, 2008.

[140] VMware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. `http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf`, 2015.

[141] VMware. VMware Workstation. `https://www.vmware.com/products/workstation`, 2015.

[142] Bing Wei, Chuang Lin, and Xiangzhen Kong. Energy optimized modeling for live migration in virtual data center. In *Proceedings of the International Conference on Computer Science and Network Technology (ICCSNT)*, pages 2311–2315, 2011.

[143] Sanford Weisberg. *Applied Linear Regression, 3rd edition*. Wiley Series in Probability and Statistics, 2005.

[144] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black–box and gray–box strategies for virtual machine migration. In *Proceedings of the 4th USENIX conference on Networked systems design and implementation*, pages 229–242, 2007.

[145] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions of Parallel and Distributed Systems*, 24:1107–1116, 2013.

[146] Fei Xu, Fangming Liu, Linghui Liu, Hai Jin, Bo Li, and Baochun Li. iaware: Marking live migration of virtual machines interference–aware in the cloud. *IEEE Transaction on Computers*, 63:3012–3025, 2014.

[147] Chih Chiang Yang, Kun Ting Chen, Chien Chen, and Jing Ying Chen. Market–based load balancing for distributed heterogeneous multi–resource servers. In *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 158–165, 2009.

[148] Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, and Bei Wang. Live migration of multiple virtual machines with resource reservation in cloud computing environments. In *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD)*, pages 267–274, 2011.

[149] Lei Ye. *Energy Management for Virtual Machines*, page 102 pages. PhD Thesis. The University of Arizona, 2013.

[150] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state–of–the–art and research challenges. *Journal of Internet Services and Applications*, 1:7–18, 2010.

[151] Jinzy Zhu. *Cloud Computing Technologies and Application*, pages 21–45. Hanbook of Cloud Computing, 2010.

Aalto-DD 140/2016

ISBN 978-952-60-6913-5 (printed)
ISBN 978-952-60-6912-8 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL DISSERTATIONS